LLNL-JRNL-669889

# Fully-Implicit Orthogonal Reconstructed Discontinuous Galerkin for Fluid Dynamics with Phase Change

R. Nourgaliev, H. Luo, B. Weston, A. Anderson, S. Schofield, T. Dunn, J. Delplanque

April 20, 2015

Journal of Computational Physics

# Fully-Implicit Orthogonal Reconstructed Discontinuous Galerkin Method for Fluid Dynamics with Phase Change

R. Nourgaliev*[a], H. Luo[b], B. Weston[c], A. Anderson[a], S. Schofield[a], T. Dunn[a], J.-P. Delplanque[c]

[a] *Lawrence Livermore National Laboratory, Design Physics Division, Livermore, CA 94551, USA*
[b] *North Carolina State University, Mechanical & Aerospace Engineering, Raleigh, NC 27695-7910, USA*
[c] *University of California Davis, Mechanical & Aerospace Engineering, Davis, CA 95616, USA*

### Abstract

A new reconstructed Discontinuous Galerkin (rDG) method, based on orthogonal basis/test functions, is developed for fluid flows on unstructured meshes. Orthogonality of basis functions is essential for enabling robust and efficient fully-implicit Newton-Krylov based time integration. The method is designed for generic partial differential equations, including transient, hyperbolic, parabolic or elliptic operators, which are attributed to many multiphysics problems. We demonstrate the method's capabilities for solving compressible fluid-solid systems (in the low Mach number limit), with phase change (melting/solidification), as motivated by applications in Additive Manufacturing (AM). We focus on the method's accuracy (in both space and time), as well as robustness and solvability of the system of linear equations involved in the linearization steps of Newton-based methods. The performance of the developed method is investigated for highly-stiff problems with melting/solidification, emphasizing the advantages from tight coupling of mass, momentum and energy conservation equations, as well as orthogonality of basis functions, which leads to better conditioning of the underlying (approximate) Jacobian matrices, and rapid convergence of the Krylov-based linear solver.

*Key words:* Discontinuous Galerkin Method, High-Order Space-Time Discretization, Fully-Implicit Solvers, Newton-Krylov Algorithm, Fluid Dynamics, Phase-Change

## 1. Introduction

In this work, we developed a new fully-implicit solution algorithm, based on the reconstructed Discontinuous Galerkin (rDG) method. The main technical contribution is the combination of the in-cell and inter-cell reconstructions using orthogonal basis/test functions on unstructured meshes, which are aimed at better conditioning of linear steps during the Newton-based non-linear iterative procedure in fully-implicit solver. In addition, we demonstrated an ability to solve for primitive variables, chosen on the requirement of solvability (better conditioning) of the underlying physics. This is in contrast to our previous rDG efforts [31–36, 38–40, 58–60], which used conservation variables as a solution vector. In the context of the application of interest, we are looking for **all-speed flow** capabilities, with **phase change (melting/solidification)**. In these cases, the set of conservation variables (i.e., mass, momentum and total energy) is poorly conditioned and restrictive in terms of feasible flow regimes (Mach number limitations). Thus, we use the sets of primitive variables, which boost solvability, such as pressure, velocity and either specific internal energy, enthalpy or temperature, as an energy transport variable. Note that residuals are always formed for mass, momentum and energy, to ensure conservation upon convergence of the non-linear solver, within the framework of the Newton-Krylov algorithm.

This work is motivated by applications in *Additive Manufacturing (AM)*. AM requires the resolution of laser-induced powder melting and the subsequent formation of liquid metal pools, with numerous computationally challenging issues [26]. These include modeling of liquid-solid interfaces, due to powder melting/solidification; gas-solid and gas-liquid multi-material interfaces, with representation of ambient gaseous media (air, Argon, Helium), using

*Corresponding author, nourgaliev1@llnl.gov.

fully-compressible formulation; and complex interfacial physics, including surface tension and Marangoni convection, adequate representation of fluid-solid-gas contacts, wetting phenomena, and metal evaporation/condensation, with associated recoil pressure effects, etc. In this study, we will focus on all-speed flow capabilities and adequate modeling of fluid-solid phase change.

To enable the numerical resolution of all-speed flow regimes, and stiffness associated with material strength models, covering both liquid (Newtonian fluid viscous stress tensors) and solids (with appropriate stress deviators, plastic strain and other constitutive models), we decided to work with fully-implicit solvers, which are based on tight coupling of all involved physics. While we are interested in resolving slow dynamic time scales of the process (mostly at the material velocity time scale), fast (stiff) time scales due to acoustics and material strength are also properly accounted for.

We capitalize on recent developments in *L*-stable time integrators [7, 8] and Newton based multiphysics algorithms [27, 48], which is in contrast to Picard-iteration based fully-implicit solution algorithms[1] [47, 49], currently widely used in commercial fluid dynamics codes, such as ANSYS-CFX, STAR-CCM+, Fluent, etc., and many research and open-source codes, such as OpenFOAM.

For space discretization, we will use the recently developed reconstructed Discontinuous Galerkin method, which has been been shown to be a highly attractive approach for high-order numerical discretization of multi-physics problems [31–36, 38–40, 44, 59, 60]. Our emphasis here is placed on making rDG work robustly within the fully-implicit framework, especially at the regimes when the underlying linear algebra is highly stiff. For this purpose, we adopt orthogonal basis functions, which support better conditioning of the time integrators. To ensure orthogonality, we use the modal DG with Legendre-based tensor-product basis functions, similar to what was used in recovery DG on two-dimensional uniform meshes in [45, 46][2]. In addition, we modify the test functions, by inverse-Jacobian-weighting the basis functions, which places the method within the general class of Petrov-Galerkin formulations. This choice of basis and test functions results in diagonal mass matrices. This new rDG method has numerous attractive features. Besides *orthogonality*, the method is *hierarchical*, which enables natural *compatibility with p-refinement* and provides means for computing *measures* to activate *h-refinement*. The method inherits all advantages from the original rDG, including an easy implementation on *hybrid meshes* and an ability to *work naturally with AMR*. Since the base (zeroth-order) degrees of freedom are cell-averaged quantities, the method should be viewed as a DG-based extension of the second-order finite-volume (FV) algorithm, to enable high-order (> $2^{\text{nd}}$) discretization on unstructured hybrid meshes, without extension of the stencil. In fact, the stencil of the rDG is exactly the same as in the second-order finite-volume methods, which are used in most commercial CFD packages. Close relationship with FV is very attractive for CFD practitioners, as most successful CFD codes are based on the FV framework.

The method is implemented and tested within LLNL's ALE3D code [1, 2]. ALE3D is a multi-physics numerical simulation tool, focusing on modeling hydrodynamics and structured mechanics in all-speed multi-material applications. Additional ALE3D features include heat conduction, chemical kinetics and species diffusion, incompressible flow, a wide range of material models, chemistry models, multi-phase flow, and magneto-hydrodynamics for long- (implicit) and short- (explicit) time-scale applications.

The rest of this paper is organized as follows. In Section 2, we start with the description of governing equations. The new rDG with orthogonal basis/test functions is introduced in Section 3. Fully-implicit time discretization and related algorithms (Newton-Krylov solver, solving in primitive variables, preconditioning) are described in Section 4. Extensive numerical testing and demonstration of the method's performance are presented in Section 5. Finally, concluding remarks and future directions are given in Section 6.

[1] We are excluding from consideration *operator-splitting methods*, such as *projection methods* [6, 9, 19]; and *"Implicit Continuous-fluid Eulerian (ICE)"* algorithm, [21, 22]). These methods are very successful in simulations of relatively simple fluid dynamics, but are not considered to be robust-enough for application of interest in the present study.

[2] It is instructive to note that similar basis functions are also used by Dumbser et al. [14, 15, 17], in conjunction with their $\mathrm{P_n P_m}$ scheme and explicit time integrators.

## 2. Mathematical Model

The governing equations can be written in the following form:

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial}{\partial x_j}\left(\mathbf{F}_j - \mathbf{D}_j\right) = \mathbf{S} \quad j{=}1,2,3 \tag{1}$$

where $t$, $\mathbf{x} = x_j = (x, y, z)$, $\mathbf{U}$, $\mathbf{F}_j$, $\mathbf{D}_j$ and $\mathbf{S}$ are time, Cartesian coordinates, the vectors of conservative variables, hyperbolic fluxes, diffusion fluxes and sources, correspondingly. We also introduce a vector of "primitive" variables, $\mathbf{W}$, which is generally different from $\mathbf{U}$, and chosen based on the "better system conditioning" considerations, Section 4.1.4.

For the compressible Navier-Stokes equations, the conservation variables are mass, linear momentum and total energy,

$$\mathbf{U} = \left[\ \rho, \rho u, \rho v, \rho w, E\ \right]^{\mathrm{T}} \tag{2}$$

while the vectors of fluxes and sources are defined as:

$$\mathbf{F}_j = \begin{bmatrix} \rho v_j \\ \rho v_j u + P\delta_{1j} \\ \rho v_j v + P\delta_{2j} \\ \rho v_j w + P\delta_{3j} \\ \rho v_j e + v_j P \end{bmatrix},\ \mathbf{D}_j = \begin{bmatrix} 0 \\ \tau_{1j} \\ \tau_{2j} \\ \tau_{3j} \\ -q_j + v_k \tau_{jk} \end{bmatrix},\ \mathbf{S} = \begin{bmatrix} m''' \\ f_1 \\ f_2 \\ f_3 \\ q''' + f_k v_k \end{bmatrix} \tag{3}$$

In eqs.(2) and (3), $\rho$, $\mathbf{v} = v_j = (u, v, w)$, $e$, and $P$ are the density, material velocity, specific total energy, and pressure, correspondingly. Volumetric mass sources, body forces, and volumetric energy sources are denoted as $m'''$, $\mathbf{f} = f_j$ and $q'''$, respectively.

### 2.1. *Constitutive physics*

Without loss of generality, we will consider here *Newtonian fluids*, for which the viscous stress tensor is defined as [3, 28]

$$\tau_{ij} = 2\mu S_{ij} + \underbrace{\left(\varsigma - \frac{2}{3}\mu\right)}_{\lambda} \partial_k v_k \delta_{ij} \tag{4}$$

where $\mu$ and $\lambda$ are the first and second Lamé parameters, respectively, while $\varsigma$ is the bulk viscosity. The strain tensor is defined as

$$S_{ij} = \frac{1}{2}\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right) \tag{5}$$

Following *Stokes* [3], $\lambda = -\frac{2}{3}\mu$ and $\varsigma = 0$.

For heat flux, we will use *Fourier law*, defined as

$$q_j = -\kappa \frac{\partial T}{\partial x_j} \tag{6}$$

where $\kappa$ is the thermal conductivity coefficient, which is in general a function of temperature $T$.

The final constitutive relationship is due to equations of state, defining

$$\rho = \rho\,(P, \mathfrak{u}) \quad \text{and} \quad \mathfrak{u} = \mathfrak{u}\,(P, T) \tag{7}$$

where $\mathfrak{u}$ is the specific internal energy. The equation of state used in this study is described in Appendix A. Here, we will restrict ourselves to *thermally perfect* materials[3], i.e. $\mathfrak{u}\,(T)$.

[3]It is important to note that our framework is not limited to equations of state in analytic form. In practical applications, more accurate represen-

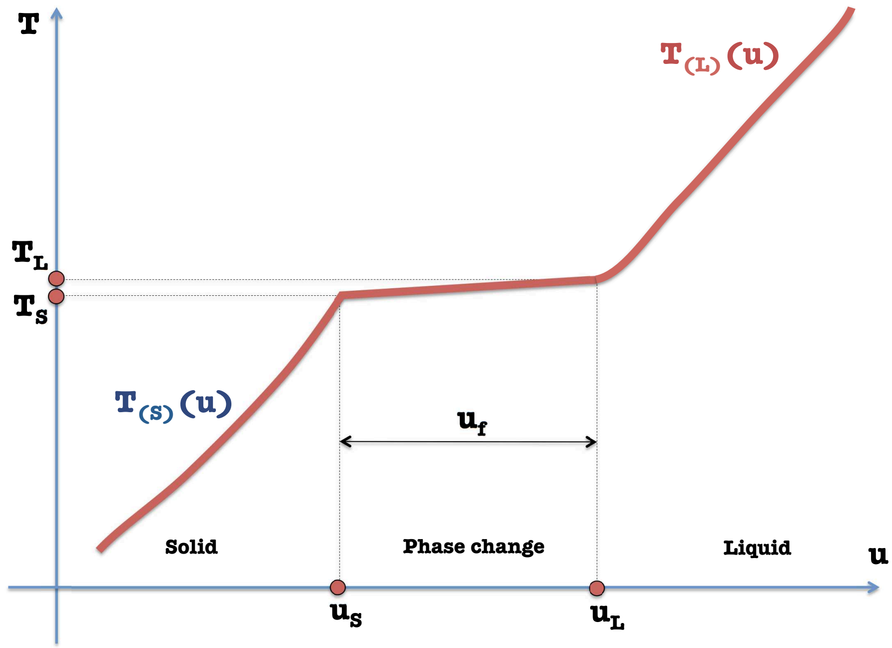

Figure 1: On thermal model with melting/solidification.

### 2.2. *Phase change*

To represent phase change, we use an *energy- (homogeneous thermal equilibrium)- based* approach. Phase transition is incorporated into the thermal model $\mathfrak{u}(T)$ as following.

Three material state zones are introduced, Figure 1.

I. **Solid.** ($T < T_S$, $\mathfrak{u} < \mathfrak{u}_S$), where $T_S$ and $\mathfrak{u}_S$ are *solidus* temperature and specific internal energy, respectively.

II. **Liquid.** ($T > T_L$, $\mathfrak{u} > \mathfrak{u}_L$), where $T_L$ and $\mathfrak{u}_L$ are *liquidus* temperature and specific internal energy, respectively.

III. **Two-phase.** ($T_S \leq T \leq T_L$, $\mathfrak{u}_S \leq \mathfrak{u} \leq \mathfrak{u}_L$).

As the materials of practical interests for AM are alloys, we always have a transitional two-phase region, between solidus and liquidus, making the mapping ($\mathfrak{u} \longleftrightarrow T$) non-singular. This enables a robust melting front capturing, with finite-thickness jumps in specific energy due to latent heat, and eliminates numerical degeneracy/oscillations when computing heat transfer across the two-phase zone.

We define the *latent heat* as

$$\mathfrak{u}_f = \mathfrak{u}_L - \mathfrak{u}_S \tag{8}$$

Thermal conductivity is represented as

$$\kappa(\mathfrak{u}) = \begin{cases} \kappa_S(T) & \text{if } \mathfrak{u} < \mathfrak{u}_S \\ \kappa_L(T_L) + \frac{\kappa_S(T_S) - \kappa_L(T_L)}{2}(1 + \cos(x\pi)) & \text{if } \mathfrak{u}_S \leq \mathfrak{u} \leq \mathfrak{u}_L \\ \kappa_L(T) & \text{otherwise} \end{cases} \tag{9}$$

---

tation is typically provided by tabulated EOSs. For instance, in [43, 61], we utilized bi-cubic spline thermodynamically-consistent representation of IAPWS-IF97 tables for water-steam systems, using a similar rDG-based Newton-Krylov framework, in applications for nuclear reactor safety.

where $\kappa_L(T)$, and $\kappa_S(T)$ are the thermal conductivity of liquid and solid, respectively, while the *thermodynamic quality* is defined as

$$x = \frac{\mathfrak{u} - \mathfrak{u}_S}{\mathfrak{u}_f} \tag{10}$$

To represent the strength of the material, in both solid state and the transitional two-phase "mushy" region, we utilize the *viscosity-based* approach[4]. It is a simple extension of the viscous stress tensor in a fluid, eq.(4), where the viscosity is set to be a function of the specific internal energy (or temperature). We use the following functional form for the dynamic viscosity:

$$\mu(T) = \begin{cases} \mu_S^\star & \text{if } T < T^\star \\ \mu_L f_\mu(T) & \text{if } T^\star \le T \le T_L \\ \mu_L & \text{otherwise} \end{cases} \tag{11}$$

where $f_\mu(T)$ is the viscosity factor, smoothly varied from 1 to a large number, $\frac{\mu_S^\star}{\mu_L}$, in the range of temperatures from liquidus $T_L$ down to $T^\star$. This model enables a smooth transition of the effective viscosity of the media from the dynamic viscosity of liquid ($\mu_L$) to the very large "solid-state" viscosity ($\mu_S^\star$), which inhibits material deformation. The viscosity factor model used here is described in Appendix B.

## 3. Reconstructed Discontinuous Galerkin (rDG)

The computational domain $\Omega$ is subdivided into a collection of non-overlapping elements, $\Omega_e$. Without loss of generality, we will consider here linear QUAD4 (4-node) and HEX8 (8-node) elements[5].

Next, let us introduce the following broken Sobolev space $\mathbb{V}_h^p$, consisting of discontinuous vector-values polynomial functions of degree $p$,

$$\mathbb{V}_h^p = \left\{ \upsilon_h \in [\mathcal{L}_2(\Omega)]^m : \upsilon_h\big|_{\Omega_e} \in \left[\mathcal{V}_p^m\right] \forall \Omega_e \in \Omega \right\} \tag{12}$$

where $m$ is the dimension of the unknown vector and $\mathcal{V}_p$ is the space of all polynomials of degree $\le p$. To formulate the basic DG method, we introduce the following weak formulation, which is obtained by multiplying eq.(1) by a test function $\mathcal{W}_h$, integrating over an element $\Omega_e$, and then performing an integration by parts,

$$\begin{aligned} \mathbf{R}_h(\mathbf{U}_h) = \frac{\partial}{\partial t}\int_{\Omega_e} \mathbf{U}_h \mathcal{W}_h d\Omega + \int_{\Gamma_e} \left(\mathbf{F}_j(\mathbf{U}_h) - \mathbf{D}_j(\mathbf{U}_h)\right) n_j \mathcal{W}_h d\Gamma - \\ - \int_{\Omega_e} \left[\left(\mathbf{F}_j(\mathbf{U}_h) - \mathbf{D}_j(\mathbf{U}_h)\right) \frac{\partial \mathcal{W}_h}{\partial x_j} + \mathbf{S}(\mathbf{U}_h)\mathcal{W}_h\right] d\Omega, \quad \forall \mathcal{W}_h \in \mathbb{V}_h^p \end{aligned} \tag{13}$$

where $\mathbf{U}_h$ and $\mathcal{W}_h$ are represented by piecewise-polynomial functions of degrees $p$, which are discontinuous between the cell interfaces, and $\mathbf{n} = n_j$ denotes the unit outward normal vector to the element face $\Gamma_e$ (i.e., the boundary of $\Omega_e$). The local residual function $\mathbf{R}_h(\mathbf{U}_h)$ defines an inner product of the solution residue representation (with a chosen set of basis functions) and the test functions $\mathcal{W}_h$. In our solution procedure, we are minimizing this inner product, making this approach a particular case of the more general class of the method of *Mean Weighted Residuals* (MWR).

Since the numerical solution $\mathbf{U}_h$ is discontinuous across element interfaces, the numerical hyperbolic fluxes are not uniquely defined. The hyperbolic flux function $\mathbf{F}_j(\mathbf{U}_h)n_j$ appearing in the face integral term of eq.(13) is replaced

[4]Another commonly-used approach is due to Voller and Prakash [57], and it is based on the adaptation of the *Darcy law*. Pros and cons of these models are discussed by Dantzig in [11]. Here, we chose to use the viscosity-based approach, as it is more numerically challenging, incorporating the non-linearity and the momentum-energy equation coupling into the diffusion operator, which highlights the overall robustness of our numerical solver.

[5]Extensions to quadratic QUAD8 and HEX20 elements, as well as to triangle and tetrahedral elements are straightforward.

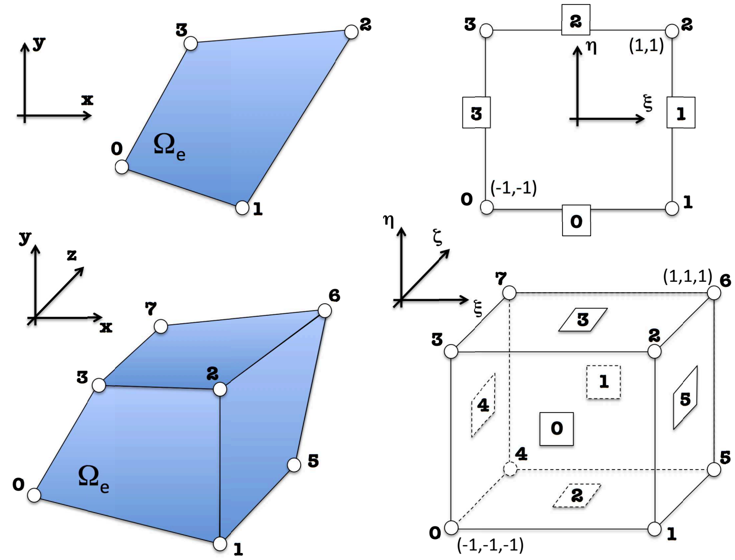


Figure 2: Element topology and isoparametric mapping for linear QUAD4 (top) and HEX8 (bottom) elements.

by a numerical Riemann flux function, $\mathbf{H}_j\left(\mathbf{U}_h^{\mathrm{L}}, \mathbf{U}_h^{\mathrm{R}}\right) n_j$, which is computed by some (approximate) Riemann solver. Here, $\mathbf{U}_h^{\mathrm{L}}$ and $\mathbf{U}_h^{\mathrm{R}}$ are the conservative state vectors at the left and right side of the element boundary. In this sense, the discontinuous Galerkin formulation is very similar to finite-volume schemes, and the DG methods can be regarded as a natural generalization of finite-volume methods to higher order.

### 3.1. *Basis and test functions*

Numerical polynomial solutions $\mathbf{U}_h$ in each element are expressed using a chosen set of basis functions $\mathcal{B}_{(k)}(\mathbf{x})$, as

$$\mathbf{U}_h(\mathbf{x}, t) = \sum_{k=0}^{K-1} \mathbf{U}_{(k)_e}(t)\, \mathcal{B}_{(k)}(\mathbf{x}) \tag{14}$$

where $\mathbf{U}_{(k)_e}$ denotes degrees of freedom (DoF) in an element $e$. In our previous work, [31–36, 38–40, 59, 60], we utilized the *Taylor-series-based* basis functions. In 2D, these are

$$\boldsymbol{\mathcal{T}}(\mathbf{x}) = \left\{ \begin{array}{l} 1,\ \underbrace{\dfrac{x-x_c}{\Delta x}}_{\mathcal{T}_{(1)}},\ \underbrace{\dfrac{y-y_c}{\Delta y}}_{\mathcal{T}_{(2)}},\ \underbrace{\dfrac{\mathcal{T}_{(1)}^2}{2} - \dfrac{1}{\Omega_e}\displaystyle\int_{\Omega_e} \dfrac{\mathcal{T}_{(1)}^2}{2} d\Omega}_{\mathcal{T}_{(3)}}, \\[2em] \underbrace{\mathcal{T}_{(1)}\mathcal{T}_{(2)} - \dfrac{1}{\Omega_e}\displaystyle\int_{\Omega_e} \mathcal{T}_{(1)}\mathcal{T}_{(2)} d\Omega}_{\mathcal{T}_{(4)}},\ \underbrace{\dfrac{\mathcal{T}_{(2)}^2}{2} - \dfrac{1}{\Omega_e}\displaystyle\int_{\Omega_e} \dfrac{\mathcal{T}_{(2)}^2}{2} d\Omega}_{\mathcal{T}_{(5)}},\ \ldots \end{array} \right\} \tag{15}$$

where $\Delta x = \frac{x_{\max} - x_{\min}}{2}$, $\Delta y = \frac{y_{\max} - y_{\min}}{2}$, and $x_{\max}$, $x_{\min}$, $y_{\max}$, and $y_{\min}$ are the maximum and minimum coordinates in the cell $\Omega_e$ in the $x$-, and $y$-directions, respectively.

Here, instead, we use tensor-product *Legendre-polynomial-based* basis functions, defined as

$$\boldsymbol{\mathcal{B}}(\xi,\eta) = \left\{ \begin{array}{l} 1,\ \underbrace{\mathfrak{L}_{(1)}(\xi)}_{\mathcal{B}_{(1)}},\ \underbrace{\mathfrak{L}_{(1)}(\eta)}_{\mathcal{B}_{(2)}},\ \underbrace{\mathfrak{L}_{(2)}(\xi)}_{\mathcal{B}_{(3)}},\ \underbrace{\mathfrak{L}_{(1)}(\xi)\,\mathfrak{L}_{(1)}(\eta)}_{\mathcal{B}_{(4)}},\ \underbrace{\mathfrak{L}_{(2)}(\eta)}_{\mathcal{B}_{(5)}}, \\ \underbrace{\mathfrak{L}_{(3)}(\xi)}_{\mathcal{B}_{(6)}},\ \underbrace{\mathfrak{L}_{(2)}(\xi)\,\mathfrak{L}_{(1)}(\eta)}_{\mathcal{B}_{(7)}},\ \underbrace{\mathfrak{L}_{(1)}(\xi)\,\mathfrak{L}_{(2)}(\eta)}_{\mathcal{B}_{(8)}},\ \underbrace{\mathfrak{L}_{(3)}(\eta)}_{\mathcal{B}_{(9)}},\ \ldots \end{array} \right\} \tag{16}$$

in 2D, and

$$\boldsymbol{\mathcal{B}}(\xi,\eta,\zeta) = \left\{ \begin{array}{l} 1,\ \underbrace{\mathfrak{L}_{(1)}(\xi)}_{\mathcal{B}_{(1)}},\ \underbrace{\mathfrak{L}_{(1)}(\eta)}_{\mathcal{B}_{(2)}},\ \underbrace{\mathfrak{L}_{(1)}(\zeta)}_{\mathcal{B}_{(3)}},\ \underbrace{\mathfrak{L}_{(2)}(\xi)}_{\mathcal{B}_{(4)}},\ \underbrace{\mathfrak{L}_{(2)}(\eta)}_{\mathcal{B}_{(5)}},\ \underbrace{\mathfrak{L}_{(2)}(\zeta)}_{\mathcal{B}_{(6)}}, \\ \underbrace{\mathfrak{L}_{(1)}(\xi)\,\mathfrak{L}_{(1)}(\eta)}_{\mathcal{B}_{(7)}},\ \underbrace{\mathfrak{L}_{(1)}(\xi)\,\mathfrak{L}_{(1)}(\zeta)}_{\mathcal{B}_{(8)}},\ \underbrace{\mathfrak{L}_{(1)}(\eta)\,\mathfrak{L}_{(1)}(\zeta)}_{\mathcal{B}_{(9)}},\ \ldots \end{array} \right\} \tag{17}$$

in 3D. In eqs.(16) and (17), $\mathfrak{L}_{(n)}(\mathrm{x})$ are Legendre polynomials of order $n$, and we utilize the isoparametric mapping from the physical space $\mathbf{x} = (x, y, z)$ to the reference space $\boldsymbol{\chi} = (\xi, \eta, \zeta)$ (Figure 2), using canonical shape functions (see [29] for details). It is instructive to note that, with these basis functions, the first degree of freedom is the *cell-averaged* quantity, which naturally connects this method to finite-volume methods. In addition, these basis functions are *modal* and *hierarchical*.

The test functions are defined as[6]

$$\mathcal{W}_{(n)} = \frac{\mathcal{B}_{(n)}}{|\mathbb{J}|} \tag{18}$$

where $\mathbb{J}$ is the *Jacobian matrix*, defined as

$$\mathbb{J} = \begin{bmatrix} x_\xi & x_\eta & x_\zeta \\ y_\xi & y_\eta & y_\zeta \\ z_\xi & z_\eta & z_\zeta \end{bmatrix} \quad \text{and} \quad \mathbb{J}^{-1} = \begin{bmatrix} \xi_x & \xi_y & \xi_z \\ \eta_x & \eta_y & \eta_z \\ \zeta_x & \zeta_y & \zeta_z \end{bmatrix} \tag{19}$$

(see [29] for details). Importantly, the basis functions eqs.(16) and (17) are orthogonal relative to the test functions eq.(18). Thus,

$$\int_{\Omega_e} \mathcal{B}_{(n)} \mathcal{W}_{(k)} d\Omega = \frac{1}{\mathcal{A}_{(n)}} \delta_{k,n} \tag{20}$$

where the *normalization coefficients* are

$$\begin{array}{ll} \mathcal{A}_{(n)} = \left\{\frac{1}{4}, \frac{3}{4}, \frac{3}{4}, \frac{5}{4}, \frac{9}{4}, \frac{5}{4}, \frac{7}{4}, \frac{15}{4}, \frac{15}{4}, \frac{7}{4}, \ldots\right\}, & \text{(QUAD)} \\ \mathcal{A}_{(n)} = \left\{\frac{1}{8}, \frac{3}{8}, \frac{3}{8}, \frac{3}{8}, \frac{5}{8}, \frac{5}{8}, \frac{5}{8}, \frac{9}{8}, \frac{9}{8}, \frac{9}{8}, \frac{7}{8}, \frac{7}{8}, \frac{7}{8}, \frac{27}{8}, \frac{15}{8}, \frac{15}{8}, \frac{15}{8}, \frac{15}{8}, \frac{15}{8}, \frac{15}{8}, \ldots\right\}, & \text{(HEX)} \end{array} \tag{21}$$

This is an important feature of this formulation, different from our previous work. ***Orthogonality is advantageous in conjunction with fully-implicit time discretization***, as the mass matrix is diagonal, with the maximum contribution towards diagonal elements of the underlying linear algebra involved in linear steps of the Newton solver (see Section 4).

[6] Since the basis and test functions are different, this approach belongs to the more general class of the *Petrov-Galerkin* formulation.

*3.2. Reconstruction*

Discontinuous Galerkin methods provide an elegant way to build high-order space discretization on unstructured meshes, by simply adding additional degrees of freedom per element, per variable. This approach is compact, i.e. DG($P_p$) of any order $p$ has the same stencil, which is a very attractive feature in terms of parallelization and code design. On the other hand, the size of the solution vector is growing significantly, as more equations for DoFs must be solved for. In particular, for modal DG in 2D – the size of the solution vector for the second- DG($P_1$), the third- DG($P_2$) and the fourth-order DG($P_3$) schemes are $\times 3$, $\times 6$ and $\times 10$ times larger, respectively, as compared to the basic DG($P_0$) finite-volume method. This is even worse in 3D, i.e. $\times 4$, $\times 10$ and $\times 20$ – for the DG($P_1$), DG($P_2$) and DG($P_3$), correspondingly. Such an increase in the size of the solution vector is unfavorable in the context of implicit solvers, imposing significant memory requirements (for storage of the linear algebra matrices) and adversely affecting solution scalability, as a majority of linear solvers do not scale linearly [53].

There is an alternative way to increase the order of accuracy, which is routine in finite-volume methods. All practically used FV methods *reconstruct in-cell slopes* of the solution vector by means of the *solution reconstruction*. This leads to an increase of the stencil, which is a well-established, acceptable strategy for the second-order FV methods. However, going beyond the second order is impractical for unstructured meshes, leading to significant complications in the code design.

In [45, 46], we explored the ideas of *recovery* to enhance the order of accuracy for DG, without increasing the size of the solution vector. We capitalized on the earlier work by van Leer and Nomura [56], who suggested to use recovery in order to consistently discretize the diffusion operator within the DG framework. In [56], the DG's piecewise discontinuous representation of the solution at the element interfaces was replaced by a *locally recovered underlying smooth solution with sufficient fidelity*, at the union of face-neighboring elements. During the recovery, it is required that the solution be consistent with the underlying DG's broken Sobolev space solution representation. We denote this type of recovery as **inter-cell** recovery.

In [46], we utilized a similar strategy[7] to enhance **in-cell** solution representation, by considering an element and its neighbors. In 1D, this leads to a family of the $\mathrm{rDG}_{P_nP_{3n+2}}$ schemes. Combined with inter-cell recovery, the $\mathrm{rDG}_{P_nP_{3n+2}}$ offers a very useful discretization framework for modeling one-dimensional fluid flows in nuclear reactor safety system analysis codes [61].

Extending the in-cell recovery to 2D regular meshes was explored in [45]. We demonstrated feasibility of the sixth-order $\mathrm{rDG}_{P_1P_5}$ for the compressible Navier-Stokes equations and radiation hydrodynamics. However, using recovery on arbitrary meshes is impractical, as it requires involvement of vertex neighbors. A more fruitful approach was explored by Dumbser et al. [14, 15, 17], who utilized *in-cell least-squares recovery*, to achieve higher order representation with DG methods. The method is called $P_nP_m$ – i.e., "*solving for* $P_n$" and "*reconstructing to* $P_m$" – a convenient way to reflect what we attempt to achieve with in-cell recovery.

As we mentioned above, recovery is not a practical approach for increasing the order of the DG method on arbitrary unstructured meshes. In [33–40, 59, 60], we utilized the **least-squares based reconstruction** instead. In this approach, we increased the accuracy of the DG method by one order, using only face-neighboring elements. This is an important design requirement, necessary for feasibility of discretization on arbitrary (hybrid) unstructured meshes. The distinguishable feature of this $\mathrm{rDG}_{P_nP_{n+1}}$ method is the use of *strong* statements, rather than *weak* (integral) statements of the recovery schemes. This strategy greatly simplifies the algorithm, and is along the lines of traditional reconstruction with finite-volume methods. In fact, the $\mathrm{rDG}_{P_0P_1}$ is exactly what is known as the least-squares based finite-volume method [5]. The $\mathrm{rDG}_{P_nP_{n+1}}$ is designed to work with modal DG, capitalizing on the availability of the solution derivatives at element centers. Thus, the in-cell higher order DoFs are reconstructed hierarchically, on the top of the solved-for DoFs, with the requirement to be consistent with the cell-centered solutions (and all available solved-for derivatives) in all face-neighboring elements, satisfied in the least-squares sense. It is easy to show that the

[7] It is instructive to note that similar recovery strategy was used by Qui and Shu [51], but in different context, building WENO limiters for DG.

method is $(n+1)$-consistent. The method is constructed in conjunction with the *inter-cell reconstruction*, which considers all unions of face-neighboring elements, and imposes the integral consistency of the face-centered reconstructed solution for cell-averaged quantities, and the least-squares-based strong statements for the solved-for cell-centered solutions and all available derivatives. This combination of the *in-cell* and *inter-cell reconstructions* provides a flexible discretization framework for generic PDEs of type eq.(1), which includes transient hyperbolic, parabolic or elliptic operators. The implementation of solution limiting, necessary for shock dynamics and multi-material applications, is done within the hierarchical WENO concept, as demonstrated in [39, 40, 60].

It must be noted that the reconstructed $\mathrm{rDG}_{\mathrm{P_n P_{n+1}}}$ suffers from the same non-orthogonality issue, as its base modal DG – i.e., the mass matrix is non-diagonal. In the following sections, we describe the $\mathrm{rDG}_{\mathrm{P_n P_{n+1}}}$ based on orthogonal basis functions (section 3.1), as well as other practical implementation issues, raised when building upon the Newton-Krylov based fully-implicit solution strategy (Section 4).

*3.2.1. In-cell reconstruction*

The basic idea of the in-cell reconstruction is to build (reconstruct) additional degrees of freedom locally, at each element, thereby increasing the order of accuracy without solving for corresponding PDEs. Consider for example the $\mathrm{rDG}_{\mathrm{P_2 P_3}}$, in two dimensions[8]. We solve for six degrees of freedom per variable (cell-average, 2 slopes and 3 curvatures), which represent the third-order-accurate piecewise-quadratic solution in the element. We want, however, to enhance the solution locally to the fourth-order, representing the piecewise-cubic solution in an element as

$$\begin{aligned} \mathbf{U}(\xi,\eta) \quad = \quad & \underbrace{\mathbf{U}_{(0)} + \mathbf{U}_{(1)}\mathcal{B}_{(1)} + \mathbf{U}_{(2)}\mathcal{B}_{(2)} + \mathbf{U}_{(3)}\mathcal{B}_{(3)} + \mathbf{U}_{(4)}\mathcal{B}_{(4)} + \mathbf{U}_{(5)}\mathcal{B}_{(5)}}_{\text{Solved for}} + \\ & \underbrace{+\mathbf{U}_{(6)}\mathcal{B}_{(6)} + \mathbf{U}_{(7)}\mathcal{B}_{(7)} + \mathbf{U}_{(8)}\mathcal{B}_{(8)} + \mathbf{U}_{(9)}\mathcal{B}_{(9)}}_{\textbf{Reconstructed}} + O\left(h^4\right) \end{aligned} \tag{22}$$

where $h$ denotes the mesh size. In this case, four degrees of freedom $\mathbf{U}_{(6,7,8,9)}$ are reconstructed in each element, using the following *least-squares in-cell reconstruction procedure*.

We seek for the missing DoFs, forming the following *normal least-squares* problem:

$$\underbrace{\begin{bmatrix} \mathcal{M}_{11_\mathrm{A}} & \mathcal{M}_{12_\mathrm{A}} & \mathcal{M}_{13_\mathrm{A}} & \mathcal{M}_{14_\mathrm{A}} \\ \mathcal{M}_{11_\mathrm{B}} & \mathcal{M}_{12_\mathrm{B}} & \mathcal{M}_{13_\mathrm{B}} & \mathcal{M}_{14_\mathrm{B}} \\ \mathcal{M}_{11_\mathrm{C}} & \mathcal{M}_{12_\mathrm{C}} & \mathcal{M}_{13_\mathrm{C}} & \mathcal{M}_{14_\mathrm{C}} \\ \mathcal{M}_{11_\mathrm{D}} & \mathcal{M}_{12_\mathrm{D}} & \mathcal{M}_{13_\mathrm{D}} & \mathcal{M}_{14_\mathrm{D}} \\ & \multicolumn{2}{c}{\dots} & \\ \mathcal{M}_{61_\mathrm{A}} & \mathcal{M}_{62_\mathrm{A}} & \mathcal{M}_{63_\mathrm{A}} & \mathcal{M}_{64_\mathrm{A}} \\ \mathcal{M}_{61_\mathrm{B}} & \mathcal{M}_{62_\mathrm{B}} & \mathcal{M}_{63_\mathrm{B}} & \mathcal{M}_{64_\mathrm{B}} \\ \mathcal{M}_{61_\mathrm{C}} & \mathcal{M}_{62_\mathrm{C}} & \mathcal{M}_{63_\mathrm{C}} & \mathcal{M}_{64_\mathrm{C}} \\ \mathcal{M}_{61_\mathrm{D}} & \mathcal{M}_{62_\mathrm{D}} & \mathcal{M}_{63_\mathrm{D}} & \mathcal{M}_{64_\mathrm{D}} \end{bmatrix}}_{\mathbb{M}} \begin{bmatrix} \mathbf{U}_{(6)} \\ \mathbf{U}_{(7)} \\ \mathbf{U}_{(8)} \\ \mathbf{U}_{(9)} \end{bmatrix} = \underbrace{\begin{bmatrix} \mathcal{R}_{1_\mathrm{A}} \\ \mathcal{R}_{1_\mathrm{B}} \\ \mathcal{R}_{1_\mathrm{C}} \\ \mathcal{R}_{1_\mathrm{D}} \\ \dots \\ \mathcal{R}_{6_\mathrm{A}} \\ \mathcal{R}_{6_\mathrm{B}} \\ \mathcal{R}_{6_\mathrm{C}} \\ \mathcal{R}_{6_\mathrm{D}} \end{bmatrix}}_{\boldsymbol{\mathcal{R}}} \tag{23}$$

Each of these 24 equations represents a constraint, being enforced on the reconstructed solution polynomial, in the least-squares sense.

**Cell-centered constraints.** The first four equations in the system (23) are formulated by requiring the in-cell solution, eq.(22), to reconstruct the *cell-centered (point-wise) solutions* in the four[9] face-neighboring cells A, B, C and D, when extended beyond the cell E. The "extended" solution profile can be written as

$$\mathbf{U}(\bar{x},\bar{y}) = \mathbb{L}^{(\mathrm{P_3})}_{2\mathrm{T_E}} \begin{bmatrix} \mathbf{U}_{(0)} \\ \mathbf{U}_{(1)} \\ \dots \\ \mathbf{U}_{(9)} \end{bmatrix}_{\mathrm{E}} \begin{bmatrix} \mathcal{T}_{(0)}(\bar{x},\bar{y}) \\ \mathcal{T}_{(1)}(\bar{x},\bar{y}) \\ \dots \\ \mathcal{T}_{(9)}(\bar{x},\bar{y}) \end{bmatrix} \tag{24}$$

[8]Extension to a generic $\mathrm{rDG}_{\mathrm{P_n P_m}}$ and three dimensions is straightforward.

[9]In the discussion, we are considering QUAD elements. Extension to TRIANGLE, HEX and TETRA elements is straightforward.

where $\mathcal{T}_{(n)}$ are the Taylor basis functions, eq.(15), while $\bar{x} = (x - x_{\mathrm{E}})$ and $\bar{y} = (y - y_{\mathrm{E}})$. The matrix $\mathbb{L}^{(\mathrm{P}_3)}_{2\mathrm{T}_{\mathrm{E}}}$ executes a mapping from DoF in our Legendre-based basis functions to those in Taylor basis functions (see Appendix C.1). Thus, each of the first four equations in (23) can be formed by setting $(\bar{x}, \bar{y})$ to $(\bar{x}_{\mathrm{A}}, \bar{y}_{\mathrm{A}})$, $(\bar{x}_{\mathrm{B}}, \bar{y}_{\mathrm{B}})$, $(\bar{x}_{\mathrm{C}}, \bar{y}_{\mathrm{C}})$ and $(\bar{x}_{\mathrm{D}}, \bar{y}_{\mathrm{D}})$, and then collecting the coefficients in the front of $\mathbf{U}_{(6,7,8,9)}$ – these are the $\mathcal{M}_{1\alpha_\beta}$ in eq.(23), while the rest of the terms are stashed into the right-hand-side terms $\mathcal{R}_{1_\beta}$, where $\alpha = 1, 2, 3, 4$ and $\beta$ =A,B,C and D.

Notably, the cell-centered solutions in the neighbor cells are directly available from the solved-for DoFs, to the order of interest:

$$\begin{aligned}\mathbf{U}\left(x_\beta, y_\beta\right) = \mathbf{U}_{(0)_\beta} - \\ - \frac{1}{\Omega_\beta} \int_{\Omega_\beta} \left( \mathbf{V}_{(3)_\beta} \frac{\left(x - x_\beta\right)^2}{2} + \mathbf{V}_{(4)_\beta} \left(x - x_\beta\right)\left(y - y_\beta\right) + \mathbf{V}_{(5)_\beta} \frac{\left(y - y_\beta\right)^2}{2} \right) d\Omega + O\left(h^4\right)\end{aligned} \tag{25}$$

where by $\mathbf{V}_{(n)}$ we denote the DoFs in terms of the Taylor basis functions. Thus, $\mathbf{V}_{(3,4,5)_\beta}$ are mapped from $\mathbf{U}_{(n)_\beta}$ using eq.(64).

This assembly procedure can be easily coded for a generic set of the $\mathrm{rDG}_{\mathrm{P_n P_m}}$ reconstruction/mesh geometry.

**Slope-consistency constraints.** The next eight equations represent the requirement for the extended solution shape to be consistent with the *point-wise solution derivatives*, in the face-neighboring cells A, B, C and D. For derivatives in $x$-directions,

$$\frac{\partial}{\partial \bar{x}} \mathbf{U}(\bar{x}, \bar{y}) = \mathbb{L}^{(\mathrm{P}_3)}_{2\mathrm{T}_{\mathrm{E}}} \begin{bmatrix} \mathbf{U}_{(0)} \\ \mathbf{U}_{(1)} \\ \cdots \\ \mathbf{U}_{(9)} \end{bmatrix}_{\mathrm{E}} \frac{\partial}{\partial \bar{x}} \begin{bmatrix} \mathcal{T}_{(0)}(\bar{x}, \bar{y}) \\ \mathcal{T}_{(1)}(\bar{x}, \bar{y}) \\ \cdots \\ \mathcal{T}_{(9)}(\bar{x}, \bar{y}) \end{bmatrix} \tag{26}$$

The assembly procedure is straightforward – set $(\bar{x}, \bar{y})$ to $(\bar{x}_{\mathrm{A}}, \bar{y}_{\mathrm{A}})$, $(\bar{x}_{\mathrm{B}}, \bar{y}_{\mathrm{B}})$, $(\bar{x}_{\mathrm{C}}, \bar{y}_{\mathrm{C}})$ and $(\bar{x}_{\mathrm{D}}, \bar{y}_{\mathrm{D}})$, collect the coefficients in the front of $\mathbf{U}_{(6,7,8,9)}$ – these are the $\mathcal{M}_{2\alpha_\beta}$ in eq.(23), and collect all the rest into right-hand-side terms $\mathcal{R}_{2_\beta}$.

The cell-centered derivatives $\mathbf{V}_{(1)_\beta}$ and $\mathbf{V}_{(2)_\beta}$ in the neighbor cells are available from the solved-for DoFs, to the order of interest (the 4th), using eq.(64).

A similar procedure is applied for the slopes in the $y$-direction, to assemble $\mathcal{M}_{3\alpha_\beta}$ and $\mathcal{R}_{3_\beta}$.

**Curvature-consistency constraints.** The rest of the 12 equations are designed to enforce the *point-wise curvature consistency*. For example, for the second derivatives in $x$-direction,

$$\frac{\partial^2}{\partial \bar{x}^2} \mathbf{U}(\bar{x}, \bar{y}) = \mathbb{L}^{(\mathrm{P}_3)}_{2\mathrm{T}_{\mathrm{E}}} \begin{bmatrix} \mathbf{U}_{(0)} \\ \mathbf{U}_{(1)} \\ \cdots \\ \mathbf{U}_{(9)} \end{bmatrix}_{\mathrm{E}} \frac{\partial^2}{\partial \bar{x}^2} \begin{bmatrix} \mathcal{T}_{(0)}(\bar{x}, \bar{y}) \\ \mathcal{T}_{(1)}(\bar{x}, \bar{y}) \\ \cdots \\ \mathcal{T}_{(9)}(\bar{x}, \bar{y}) \end{bmatrix} \tag{27}$$

which can be used to assemble $\mathcal{M}_{4\alpha_\beta}$ and $\mathcal{R}_{4_\beta}$.

Similar equations are applied to assemble $\mathcal{M}_{5\alpha_\beta}$ and $\mathcal{R}_{5_\beta}$ from the cross-derivatives $\frac{\partial^2}{\partial \bar{x}\, \partial \bar{y}}$, and $\mathcal{M}_{6\alpha_\beta}$ and $\mathcal{R}_{6_\beta}$ from $\frac{\partial^2}{\partial \bar{y}^2}$.

The cell-centered curvatures $\mathbf{V}_{(3)_\beta}$, $\mathbf{V}_{(4)_\beta}$ and $\mathbf{V}_{(5)_\beta}$ in the neighbor cells are available from the solved-for DoFs, to the 4th order of interest, using eq.(64).

**Least-Squares Solution.** Once the non-square matrix $\mathbb{M}$ and the *r.h.s.* vector $\boldsymbol{\mathcal{R}}$ are assembled, the least-squares problem, eq.(23), can be solved for $\mathbf{U}_{(6,7,8,9)}$ as:

$$\left[\mathbf{U}_{(6)}, \mathbf{U}_{(7)}, \mathbf{U}_{(8)}, \mathbf{U}_{(9)}\right]^{\mathrm{T}} = \left(\mathbb{M}^{\mathrm{T}} \mathbb{M}\right)^{-1} \left(\mathbb{M}^{\mathrm{T}} \boldsymbol{\mathcal{R}}\right) \tag{28}$$

Importantly, before applying eq.(28), both $\mathbb{M}$ and $\boldsymbol{\mathcal{R}}$ are normalized by first finding the element of the matrix $\mathbb{M}$ with the largest absolute value, $\mathcal{M}_{\max}$, and then dividing both $\mathbb{M}$ and $\boldsymbol{\mathcal{R}}$ by $\mathcal{M}_{\max}$. This helps to get a better conditioned matrix inversion in eq.(28), which is especially relevant for the higher-order ($> 3^{\text{rd}}$) reconstructions.

**Blended recovery/reconstruction operator.** We found that a better (more accurate) solution can be achieved by adding integral (weak) statements into the least-squares formulation eq.(23). In this case, the least-squares problem become:

$$\underbrace{\left[\begin{array}{cccc} \mathcal{M}_{11_\mathrm{A}} & \mathcal{M}_{12_\mathrm{A}} & \mathcal{M}_{13_\mathrm{A}} & \mathcal{M}_{14_\mathrm{A}} \\ & & \cdots & \\ \mathcal{M}_{61_\mathrm{D}} & \mathcal{M}_{62_\mathrm{D}} & \mathcal{M}_{63_\mathrm{D}} & \mathcal{M}_{64_\mathrm{D}} \\ \hline \mathcal{M}_{71_\mathrm{o}} & \mathcal{M}_{72_\mathrm{o}} & \mathcal{M}_{73_\mathrm{o}} & \mathcal{M}_{74_\mathrm{o}} \\ & & \cdots & \end{array}\right]}_{\mathbb{M}} \left[\begin{array}{c} \mathbf{U}_{(6)} \\ \mathbf{U}_{(7)} \\ \mathbf{U}_{(8)} \\ \mathbf{U}_{(9)} \end{array}\right] = \underbrace{\left[\begin{array}{c} \mathcal{R}_{1_\mathrm{A}} \\ \cdots \\ \mathcal{R}_{6_\mathrm{D}} \\ \hline \mathcal{R}_{7_\mathrm{o}} \\ \cdots \end{array}\right]}_{\boldsymbol{\mathcal{R}}} \tag{29}$$

where the equations

$$\mathcal{M}_{71_\mathrm{o}}\mathbf{U}_{(6)} + \mathcal{M}_{72_\mathrm{o}}\mathbf{U}_{(7)} + \mathcal{M}_{73_\mathrm{o}}\mathbf{U}_{(8)} + \mathcal{M}_{74_\mathrm{o}}\mathbf{U}_{(9)} = \mathcal{R}_{7_\mathrm{o}}, \quad \mathrm{o} = 0, ..., \mathrm{N}_7 \tag{30}$$

are added to enforce (in the least-squares sense) the integral consistency (recovery). In eq.(30), $\mathrm{N}_7$ could optionally be 0, 2 or 5, depending on the order of weak statements decided to be "blended in" (cell-average-, slope- or curvature- "integral consistency")[10].

More specifically, the integral statements being implemented are

$$\int_{\Omega_\beta} \omega_w \mathbf{U}(x,y)\, \mathcal{B}_{(\mathrm{o})_\beta}(x,y)\, d\Omega = \frac{\omega_w \mathbf{U}_{(\mathrm{o})_\beta}}{\mathcal{A}_{(\mathrm{o})}} \tag{31}$$

where the $\omega_w$ is the weight of the integral statements[11]; the $\mathbf{U}(x,y)$ is the reconstructed solution in the cell under consideration, eq.(22); the $\mathcal{B}_{(\mathrm{o})_\beta}$ is the orthogonal basis function (the $\mathrm{o}^{\text{th}}$-order) in the neighbor cell $\beta$; the $\mathcal{A}_{(\mathrm{o})}$ is the normalization coefficient, as defined by eq.(21); and the $\mathbf{U}_{(\mathrm{o})_\beta}$ is the $\mathrm{o}^{\text{th}}$-order DoF in the neighbor cell $\beta$. The eq.(31) corresponds to the constraints utilized in the *in-cell recovery schemes*, as introduced in [45, 46]. This enforces the Sobolev-space consistency of the reconstructed fields, with the solved-for DoFs – here, in the least-squares sense.

To convert eq.(31) to the discrete form eq.(30), we replace the integral with the summation over the Gauss integration points in the neighbor cells $\beta$, collect the coefficients in the front of $\mathbf{U}_{(6,7,8,9)}$ – these are the $\mathcal{M}_{7\alpha_\mathrm{o}}$, and stash all the rest terms into the *r.h.s.* terms $\mathcal{R}_{7_\mathrm{o}}$. This assembly procedure can be easily generalized to any order and in three dimensions.

### 3.2.2. Inter-cell reconstruction

The inter-cell reconstruction is applied to compute parabolic (and elliptic) operators. For these, we consider the unions of the face-neighboring cells. For each pair, we reconstruct a "consistent-with-parabolic-operator" solution profile, which is used to compute the diffusion flux at each Gauss point of the corresponding face.

Without loss of generality, we will discuss a union of cells E and A, as depicted in Figure 3. For 2D $\mathrm{rDG}_{\mathrm{P}_2\mathrm{P}_3}$, the *inter-cell reconstructed solution* is

$$\begin{aligned} \mathbf{U}(\bar{x},\bar{y}) \;=\; \mathbf{V}_{(0)} & + \mathbf{V}_{(1)}\mathcal{T}_{(1)}(\bar{x},\bar{y}) + \mathbf{V}_{(2)}\mathcal{T}_{(2)}(\bar{x},\bar{y}) + \mathbf{V}_{(3)}\mathcal{T}_{(3)}(\bar{x},\bar{y}) + \\ & + \mathbf{V}_{(4)}\mathcal{T}_{(4)}(\bar{x},\bar{y}) + \mathbf{V}_{(5)}\mathcal{T}_{(5)}(\bar{x},\bar{y}) + \mathbf{V}_{(6)}\mathcal{T}_{(6)}(\bar{x},\bar{y}) + \\ & + \mathbf{V}_{(7)}\mathcal{T}_{(7)}(\bar{x},\bar{y}) + \mathbf{V}_{(8)}\mathcal{T}_{(8)}(\bar{x},\bar{y}) + \mathbf{V}_{(9)}\mathcal{T}_{(9)}(\bar{x},\bar{y}) + O\left(h^4\right) \end{aligned} \tag{32}$$

[10] In most cases considered here, we enforce only the cell-average ($\mathrm{N}_7 = 0$) consistency.

[11] In most cases considered here, $\omega_w$ is set to 10, i.e. contributing ten times more than the strong statements.

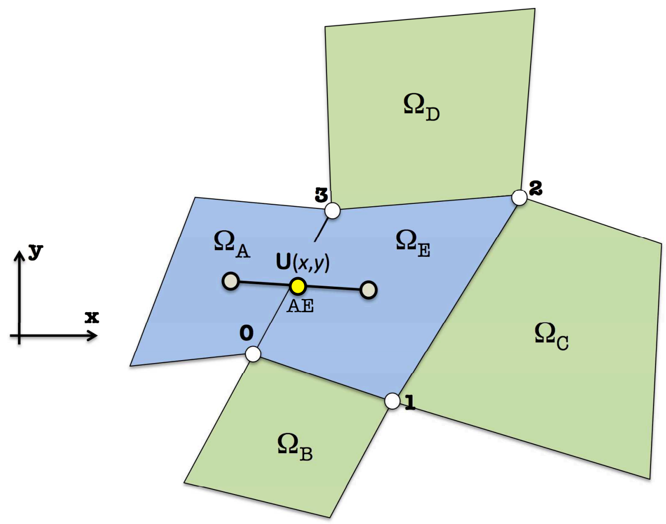


Figure 3: On the inter-cell reconstruction.

where $\bar{x} = (x - x_{AE})$, $\bar{y} = (y - y_{AE})$, and $(x_{AE}, y_{AE}) = \left(\frac{x_A + x_E}{2}, \frac{y_A + y_E}{2}\right)$. The degrees of freedom being reconstructed are $\mathbf{V}_{(0,...,9)}$. Note that these are based on the Taylor basis functions, as it is more consistent/convenient with the intended use in the diffusion operators (the Fourier fluxes and the Newtonian viscous stress tensor).

To find these DoFs, we form the following *normal least-squares* problem:

$$\underbrace{\begin{bmatrix} \mathcal{M}_{0,0_A} & \mathcal{M}_{0,1_A} & \dots & \mathcal{M}_{0,9_A} \\ \mathcal{M}_{0,0_E} & \mathcal{M}_{0,1_E} & \dots & \mathcal{M}_{0,9_E} \\ & & \dots & \\ \mathcal{M}_{9,0_A} & \mathcal{M}_{9,1_A} & \dots & \mathcal{M}_{9,9_A} \\ \mathcal{M}_{9,0_E} & \mathcal{M}_{9,1_E} & \dots & \mathcal{M}_{9,9_E} \\ \hline \mathcal{M}_{10,0_o} & \mathcal{M}_{10,1_o} & \dots & \mathcal{M}_{10,9_o} \\ & & \dots & \end{bmatrix}}_{\mathbb{M}} \begin{bmatrix} \mathbf{V}_{(0)} \\ \mathbf{V}_{(1)} \\ \dots \\ \mathbf{V}_{(9)} \end{bmatrix} = \underbrace{\begin{bmatrix} \mathcal{R}_{0_A} \\ \mathcal{R}_{0_E} \\ \dots \\ \mathcal{R}_{9_A} \\ \mathcal{R}_{9_E} \\ \hline \mathcal{R}_{10_o} \\ \dots \end{bmatrix}}_{\mathcal{R}} \tag{33}$$

The first 20 equations enforce a point-wise consistency of the inter-cell reconstructed solution with the in-cell reconstructed solutions, in the cell centers ($\mathbf{r}_A$ and $\mathbf{r}_E$). These are – two solution values, four slopes, six curvatures, and eight (reconstructed) third-order derivatives. In the generic representation, the individual equations used for the assembly are

$$\frac{\partial^{s+k}}{\partial \bar{x}^s \bar{y}^k} \mathbf{U}(\bar{x}, \bar{y}) = \begin{bmatrix} \mathbf{V}_{(0)} \\ \mathbf{V}_{(1)} \\ \dots \\ \mathbf{V}_{(9)} \end{bmatrix}_{AE} \frac{\partial^{s+k}}{\partial \bar{x}^s \bar{y}^k} \begin{bmatrix} \mathcal{T}_{(0)}(\bar{x}, \bar{y}) \\ \mathcal{T}_{(1)}(\bar{x}, \bar{y}) \\ \dots \\ \mathcal{T}_{(9)}(\bar{x}, \bar{y}) \end{bmatrix} \tag{34}$$

Similar to the procedure used for the in-cell reconstruction, we collect the coefficients in the front of $\mathbf{V}_{(0,...,9)}$ (these are the $\mathcal{M}_{\gamma,\alpha_\beta}$ in eq.(33)), and stash all the rest into right-hand-side terms $\mathcal{R}_{\gamma_\beta}$. In this case, $(\gamma, \alpha) = 0, ..., 9$, and $\beta$ =A, E.

The equations

$$\mathcal{M}_{10,0_o}\mathbf{V}_{(0)} + \mathcal{M}_{10,1_o}\mathbf{V}_{(1)} + ... + \mathcal{M}_{10,9_o}\mathbf{V}_{(9)} = \mathcal{R}_{10_o}, \quad o = 0, ..., 9 \tag{35}$$

represent the "blended-in" weak statements, which increase the accuracy and the robustness of the reconstruction. Similar to the in-cell reconstruction, we use Gaussian quadrature integration rules and the larger weights for these constraints[12].

Finally, after the non-square matrix $\mathbb{M}$ and the vector $\mathcal{R}$ are assembled, we solve for

$$\left[\mathbf{V}_{(0)}, \mathbf{V}_{(1)}, \ldots \mathbf{V}_{(9)}\right]^{\mathrm{T}} = \left(\mathbb{M}^{\mathrm{T}}\mathbb{M}\right)^{-1}\left(\mathbb{M}^{\mathrm{T}}\boldsymbol{\mathcal{R}}\right) \tag{36}$$

The matrix $\mathbb{M}$ and the vector $\mathcal{R}$ are scaled by $\mathcal{M}_{\max}$, and we use the Gauss-Jordan algorithm for the matrix inversion.

### 3.3. Boundary conditions

Boundary elements require special consideration. In this case, some of the neighbor elements are missing. While it is totally acceptable to just drop corresponding equations from the least-squares formulations (the resulting linear algebra is still well-behaved), we do enforce boundary conditions in the reconstruction, as described in [44]. We believe that infusing BC into the reconstruction results in a better-conditioned and more accurate discretization scheme. For the sake of brevity, we refer to [44] for more detail description and explanations.

### 3.4. Putting all together

The weighted residual vector for each finite element can be expressed as

$$\begin{aligned}\boldsymbol{\mathcal{R}}_{(k)} = \quad \partial_t \mathbf{U}_{(k)} \quad & + \mathcal{A}_{(k)} \int_{\Gamma_e} \left(\mathbf{F}_j - \mathbf{D}_j\right) n_j \frac{\mathcal{B}_{(k)}}{|\mathbb{J}|} d\Gamma \; - \\ & \underbrace{- \mathcal{A}_{(k)} \int_{\Omega_e} \left[\left(\mathbf{F}_j - \mathbf{D}_j\right)\left(\partial_j \mathcal{B}_{(k)} - \frac{\mathcal{B}_{(k)}}{|\mathbb{J}|}\partial_j |\mathbb{J}|\right) + \mathbf{S}\mathcal{B}_{(k)}\right] \frac{1}{|\mathbb{J}|} d\Omega}_{\boldsymbol{\mathcal{S}}(\mathbf{U})}\end{aligned} \tag{37}$$

where $k$ denotes the DoFs we solve for, and

$$\partial_j \mathcal{B}_{(k)} = \frac{\partial \mathcal{B}_{(k)}}{\partial \xi}\frac{\partial \xi}{\partial x_j} + \frac{\partial \mathcal{B}_{(k)}}{\partial \eta}\frac{\partial \eta}{\partial x_j} + \frac{\partial \mathcal{B}_{(k)}}{\partial \zeta}\frac{\partial \zeta}{\partial x_j}, \quad x_j = \{x, y, z\} \tag{38}$$

$$\partial_j |\mathbb{J}| = \frac{\partial |\mathbb{J}|}{\partial \xi}\frac{\partial \xi}{\partial x_j} + \frac{\partial |\mathbb{J}|}{\partial \eta}\frac{\partial \eta}{\partial x_j} + \frac{\partial |\mathbb{J}|}{\partial \zeta}\frac{\partial \zeta}{\partial x_j} \tag{39}$$

The discrete forms of these residuals will appear in the non-linear solver, as described in Section 4.

The *face*- and *domain*-integrals, $\int_{\Gamma_e}$ and $\int_{\Omega_e}$, correspondingly, are replaced by the weighted summation over the corresponding Gauss points. The number of quadrature points used is chosen to integrate exactly polynomials of the $(2n)^{\text{th}}$ and the $(2n+1)^{\text{th}}$ order for volume and surface inner products in the reference element, respectively, with the $n$ denoting the order of the underlying DG.

The hyperbolic fluxes $\mathbf{F}_j$ are computed using piecewise-discontinuous solutions at each face Gauss point, evaluated using the *in-cell reconstructed* solution profiles (Section 3.2.1), from each side of the face. Using these solutions, $\mathbf{U}^{(g,L)}$

[12] Mostly, we set the weight to 10.

and $\mathbf{U}^{(g,R)}$, an approximate Riemann solver is applied to compute the numerical flux. For the test cases with relatively large Mach numbers, we use the **Local Lax Friedrichs** (LLF) scheme [55]:

$$\mathbf{F}_j^{(g)} = \frac{1}{2}\left(\mathbf{F}_j^{(g,L)} + \mathbf{F}_j^{(g,R)} + \alpha_{LLF}\lambda\left(\mathbf{U}^{(g,L)} - \mathbf{U}^{(g,R)}\right)\right) \tag{40}$$

where $\lambda$ is the maximum eigenvalue, while $\alpha_{LLF}$ is the adjustable numerical diffusion coefficient – usually set to 1.

For the Navier-Stokes solver with very low Mach ($M$) numbers, we either set the coefficient $\alpha_{LLF}$ to small numbers (say, for $M = 10^{-2}$, $\alpha_{LLF} = 0.1$), or use the following "**incompressible**" variation:

$$\mathbf{F}_j^{(g)} = \frac{1}{2}\left(\mathbf{F}_j^{(g,L)} + \mathbf{F}_j^{(g,R)} + |\mathbf{v}|_{max}\left(\mathbf{U}^{(g,L)} - \mathbf{U}^{(g,R)}\right)\right) \tag{41}$$

where $|\mathbf{v}|_{max}$ is the input parameter, corresponding to the maximum material velocity for the problem under consideration.

For evaluation of the diffusion fluxes, we utilize the *inter-cell reconstructed* solution profiles (Section 3.2.2), to compute the corresponding numerical diffusion flux at each face Gauss integration point. Here, we do not consider any jumps in material properties, though we allow non-linearity of the diffusion coefficients (i.e., temperature-dependent).

The fluxes and source terms appearing in the domain integral of eq.(37) are evaluated using the *in-cell reconstructed* solution profiles. Because we are using the in-cell reconstructed solution, there is no need for modifications as discussed in [30], referred to as RDG-2x (twice-partially-integrated), including extra boundary terms. Rather, our approach is along the lines of the RDG-1x (once-partially-integrated), which offers an easy implementation of non-linear diffusion operators.

## 4. Fully-Implicit Time Discretization

An implicit time discretization of Eq.(1) can be written as

$$\begin{aligned} \mathbf{U}^{[k]} &= \mathbf{U}^{[n]} + \Delta t \sum_{r=1}^{k} \mathsf{a}_{kr} \boldsymbol{\mathcal{S}}\left(\mathbf{U}^{[r]}\right), \quad k = 1, ..., s-1 \\ \mathbf{U}^{[n+1]} &= \alpha \mathbf{U}^{[n-1]} + \beta \mathbf{U}^{[n]} + \Delta t \left(\mathsf{b}_0 \boldsymbol{\mathcal{S}}\left(\mathbf{U}^{[n]}\right) + \sum_{r=1}^{s} \mathsf{b}_r \boldsymbol{\mathcal{S}}\left(\mathbf{U}^{[r]}\right)\right) \end{aligned} \tag{42}$$

where $s$ is the total number of implicit Runge-Kutta (IRK) stages, while $\mathsf{a}_{kr}$ and $\mathsf{b}_r$ are the stage and the main scheme weights, respectively. Particular cases of interest here are:

**$\mathrm{BE}_1$** The first-order *Backward Euler* scheme, $\alpha = 0, \beta = 1, s = 1, \mathsf{b}_0 = 0$ and $\mathsf{b}_1 = 1$.

**$\mathrm{BDF}_2$** The second-order *Backward Difference* scheme, $\alpha = -\frac{\Delta t^2}{\Delta t_{n-1}(2\Delta t + \Delta t_{n-1})}$, $\beta = \frac{\Delta t}{\Delta t_{n-1}} \frac{\Delta t + \Delta t_{n-1}}{2\Delta t + \Delta t_{n-1}}$, $s = 1$, $\mathsf{b}_0 = 0$ and $\mathsf{b}_1 = 1$.

**$\mathrm{CN}_2$** The second-order *Crank-Nicholson* scheme, $\alpha = 0, \beta = 1, s = 1, \mathsf{b}_0 = \frac{1}{2}$ and $\mathsf{b}_1 = \frac{1}{2}$.

**$\mathrm{ESDIRK}_p$** The $p^{\text{th}}$-order *Explicit, Singly-Diagonal Implicit Runge-Kutta* schemes, defined by $\mathsf{b}_0 = 0$ and the Butcher tableau of the following form:

$$\begin{array}{c|cccccc} 0 & 0 & 0 & 0 & 0 & ... & 0 \\ \mathsf{c}_2 & \mathsf{a}_{21} & \gamma & 0 & 0 & ... & 0 \\ \mathsf{c}_3 & \mathsf{a}_{31} & \mathsf{a}_{32} & \gamma & 0 & ... & 0 \\ ... & & & ... & & & \\ \mathsf{c}_{s-1} & \mathsf{a}_{(s-1)1} & \mathsf{a}_{(s-1)2} & ... & ... & \gamma & 0 \\ 1 & \mathsf{b}_1 & \mathsf{b}_2 & \mathsf{b}_3 & ... & \mathsf{b}_{(s-1)} & \gamma \\ \hline & \mathsf{b}_1 & \mathsf{b}_2 & \mathsf{b}_3 & ... & \mathsf{b}_{(s-1)} & \gamma \\ \hline & \hat{\mathsf{b}}_1 & \hat{\mathsf{b}}_2 & \hat{\mathsf{b}}_3 & ... & \hat{\mathsf{b}}_{(s-1)} & \hat{\mathsf{b}}_{(s)} \end{array} \tag{43}$$

where $\mathsf{c}_r$ denotes the point in time of the $r^{\text{th}}$-stage, $t^{[n]} + \mathsf{c}_r \Delta t$. Note that the first stage is explicit, and the diagonal elements for all stages $r > 1$ are the same, $\mathsf{a}_{rr} = \gamma$. Note that the scheme is embedded, i.e., the $p^{\text{th}}$-order ESDIRK$_p$ scheme allows to compute the $(p-1)^{\text{th}}$-order solution, as

$$\mathbf{U}^{[n+1]} \quad = \quad \mathbf{U}^{[n]} + \Delta t \sum_{r=1}^{s} \hat{\mathsf{b}}_r \boldsymbol{\mathcal{S}}\left(\mathbf{U}^{[r]}\right) \tag{44}$$

The coefficients of ESDIRK$_{3,4,5}$'s Butcher tableau were derived in [7, 8, 46], and summarized in [44].

All these schemes, with the exception of CN$_2$, are *L-stable*. CN$_2$ is *A-stable* [20].

### 4.1. Newton-Krylov (NK) solver

Each stage of the IRK eq.(42) requires a solution of the non-linear system in the form

$$r\vec{e}s\left(\vec{\mathcal{X}}\right) = 0 \tag{45}$$

where

$$\vec{\mathcal{X}} = \left(\mathbf{U}^{\mathrm{T}}_{(k=0,\dots,p)_1},\ \mathbf{U}^{\mathrm{T}}_{(k=0,\dots,p)_2},\ \dots,\ \mathbf{U}^{\mathrm{T}}_{(k=0,\dots,p)_{\mathrm{N_{cells}}}}\right)^{\mathrm{T}} \tag{46}$$

is a solution vector which includes all $(p+1)$ degrees of freedom for all variables in all $\mathrm{N_{cells}}$ computational cells[13].

#### 4.1.1. Newton's method

We solve Eq.(45) with Newton's method [27], iteratively, as a sequence of linear problems defined by

$$\mathbb{J}^{a}\, \delta\vec{\mathcal{X}}^{a} = -r\vec{e}s\left(\vec{\mathcal{X}}^{a}\right) \tag{47}$$

The matrix $\mathbb{J}^{a}$ is the Jacobian of the $a^{\text{th}}$ Newton's iteration and $\delta\vec{\mathcal{X}}^{a}$ is the update vector. Each $_{(i,j)^{\text{th}}}$ element of the Jacobian matrix is a partial derivative of the $i^{\text{th}}$ equation with respect to the $j^{\text{th}}$ variable:

$$\mathbb{J}_{i,j} \equiv \frac{\partial res_i}{\partial \mathcal{X}_j} \tag{48}$$

The linear system Eq.(47) is solved for $\delta\vec{\mathcal{X}}^{a}$, and the new Newton's iteration value for $\vec{\mathcal{X}}$ is then computed as

$$\vec{\mathcal{X}}^{a+1} = \vec{\mathcal{X}}^{a} + \lambda^{a}\, \delta\vec{\mathcal{X}}^{a} \tag{49}$$

where $\lambda^{a}$ is the step-length determined by a line search[14] procedure [13], while $\delta\vec{\mathcal{X}}^{a}$ is the search direction. Newton's iterations on $\vec{\mathcal{X}}$ are continued until the convergence criterion

$$\left\| r\vec{e}s\left(\vec{\mathcal{X}}^{a}\right) \right\|_2 < \mathrm{tol_N} \left\| r\vec{e}s\left(\vec{\mathcal{X}}^{0}\right) \right\|_2 \tag{50}$$

is satisfied. The nonlinear tolerance $\mathrm{tol_N}$ is varied from $10^{-3}$ to $\mathrm{tol_N} = 10^{-8}$.

[13] It is instructive to emphasize that the reconstructed DoFs are not a part of the solution vector.

[14] In most simulations presented here, we used either the cubic backtracking or the critical-point algorithm, as implemented in PETSC [4].

*4.1.2. Krylov subspace iterations (GMRES)*

For the linear solver, we use the Arnoldi-based *Generalized Minimal RESidual method (GMRES)* [54]. Since the GMRES does not require individual elements of the Jacobian matrix $\mathbb{J}$, the matrix never needs to be explicitly constructed. Instead only matrix-vector multiplications $\mathbb{J}\vec{\kappa}$ are needed, where $\vec{\kappa}$ are Krylov vectors. The action of the Jacobian matrix are approximated in the Jacobian-free manner, by Fréchet derivatives

$$\mathbb{J}\vec{\kappa} \approx \frac{r\vec{e}s\left(\vec{\mathcal{X}} + \varepsilon\vec{\kappa}\right) - r\vec{e}s\left(\vec{\mathcal{X}}\right)}{\varepsilon} \tag{51}$$

(see [27] for choosing $\varepsilon$). Here, we use the *inexact Newton's method* [27], solving to a tight tolerance only when the added accuracy matters – i.e., when it affects the convergence of the Newton's iterations. This is accomplished by making the convergence of the linear residual proportional to the non-linear residual:

$$\left\|\mathbb{J}^{a}\delta\vec{\mathcal{X}}^{a} + r\vec{e}s\left(\vec{\mathcal{X}}^{a}\right)\right\| \leq \nu_{a}\left\|r\vec{e}s\left(\vec{\mathcal{X}}^{a}\right)\right\| \tag{52}$$

where $\nu_a$ is a constant[15].

*4.1.3. Preconditioning GMRES*

Because GMRES stores all of the previous Krylov vectors, it is necessary to keep the number of iterations relatively small, to prevent the storage and CPU time from becoming prohibitive. This is accomplished by preconditioning the linear system. We are using the right-preconditioned form of the linear system [53],

$$\underbrace{\mathbb{J}^{a}\mathbb{P}^{-1}}_{\hat{\mathbb{J}}}\underbrace{\mathbb{P}\delta\vec{\mathcal{X}}^{a}}_{\delta\vec{\mathcal{Y}}^{a}} = -r\vec{e}s\left(\vec{\mathcal{X}}^{a}\right) \tag{53}$$

where $\mathbb{P}^{-1}$ approximates $\mathbb{J}^{-1}$. The right-preconditioned version of Eq.(51) is

$$\mathbb{J}\mathbb{P}^{-1}\vec{\kappa} \approx \frac{r\vec{e}s\left(\vec{\mathcal{X}} + \varepsilon\mathbb{P}^{-1}\vec{\kappa}\right) - r\vec{e}s\left(\vec{\mathcal{X}}\right)}{\varepsilon} \tag{54}$$

Finding a good preconditioner is often a combination of art, science, and intuition. A mathematically good preconditioner should efficiently cluster the eigenvalues of the iteration matrix [27, 53]. A detailed discussion of the preconditioning procedure $\left\{\mathbb{P}^{-1}\vec{\kappa}\right\}$ is beyond the scope of the present study. In summary, we use the *lagged-thresholded* (reduced-size) Jacobian matrix $\mathbb{J}_{\text{LT}}$ as $\mathbb{P}$. The $\mathbb{J}_{\text{LT}}$ is evaluated using the smallest-last-ordering (SLO) graph coloring algorithm [10]. The matrix inversion $\mathbb{P}^{-1}$ is done using parallel LU decomposition. We will discuss all these details in a future publication.

*4.1.4. Preconditioning with primitive variable formulation*

While the residual function eq.(13) is written to satisfy the underlying **conservation laws**, conservative variables $\mathbf{U}$ are often a poor choice to be solved for, as the Jacobian matrix $\mathbb{J}_{i,j} \equiv \frac{\partial res_i}{\partial \mathbf{U}_j}$ might be extremely ill-conditioned. This is a case for low-Mach-number and stiff compressible fluids, which are of interest here. For stiff fluids (such as water or liquid metals), small variations in density cause large variations of pressure and internal energy. On the other hand, density is rather weakly sensitive (or even completely insensitive in the incompressible limit $M \to 0$) to both pressure and internal energy. Similarly, total energy is a poor choice for the low-speed flow regime, because it is very weakly dependent on the kinetic energy, as $\frac{\mathbf{v}^2}{2} \ll \mathfrak{u}$. All these lead to degenerate (ill-conditioned) Jacobian matrices for underlying linear algebra.

One of the great strengths of the non-linear Newton-Krylov algorithm is that it is not required to solve for conservative variables. Instead, one can (should) solve for another (mathematically equivalent) set of unknowns, which will

[15] In most computations of the present study, we use $\nu_a = 10^{-3}$.

render a better conditioned system. We denote this set of unknowns as **primitive variables**, $\mathbf{W}$.

Introducing a transformation

$$\delta \mathbf{U} = \frac{\partial \mathbf{U}}{\partial \mathbf{W}} \delta \mathbf{W} \tag{55}$$

linear steps of Newton iterations (i.e., eq.(47)) can be written as:

$$\underbrace{\frac{\partial \vec{res}}{\partial \mathbf{U}} \frac{\partial \mathbf{U}}{\partial \mathbf{W}}}_{\tilde{\mathbb{J}}_{i,j} \equiv \frac{\partial res_i}{\partial \mathbf{W}_j}} \delta \mathbf{W} = -\vec{res} \tag{56}$$

Note that with a proper choice of $\mathbf{W}$, the Jacobian matrix $\tilde{\mathbb{J}}_{i,j} \equiv \frac{\partial res_i}{\partial \mathbf{W}_j}$ is better conditioned. This should be viewed as preconditioning, as it is conceptually similar to eq.(53). It is important to note that the change of variables does not affect conservation, as the residual functions are still written in the conservative form. Upon the convergence of non-linear iterations, the conservation is satisfied to the chosen tolerance level.

In most computations presented here, we use the $[P\mathbf{v}T]$-formulation:

$$\mathbf{W} = [P, u, v, w, T]^{\mathrm{T}} \tag{57}$$

i.e., the piecewise-polynomial solution representation, as well as the solved-for and reconstructed DoFs correspond to pressure, material velocity and temperature.

The main technical challenge in implementing this preconditioning – is an ability to map the solved-for DoF ($\mathbf{W}$) to those of the conservative vector $\mathbf{U}$, as these are needed for evaluation of the (weighted) undivided differences (time-derivatives) $\left(\mathbf{U}^{[k]} - \mathbf{U}^{[n]}\right)$ and $\left(\mathbf{U}^{[n+1]} - \alpha \mathbf{U}^{[n-1]} - \beta \mathbf{U}^{[n]}\right)$, in eq.(42). This mapping is described in Appendix C.2.

## 5. Numerical Experiments

In this section, we demonstrate performance of the new algorithm. We start with the discussion of the method's accuracy, consistency and convergence, using the heat diffusion test on highly distorted meshes (Kershaw problem, Section 5.1.1) and the manufactured solution for the compressible Navier-Stokes equations (Section 5.1.2). Then, we move to fluid flow problems, increasing the level of complexity, from forced convection (Section 5.2.1), to vortex-shedding flows (Section 5.2.2), then to natural convection flows (Section 5.3), finishing with melt pool dynamics with crust formation and stable/unstable stratification effects (Section 5.4). This test suite demonstrates a wide range of the method's applicability.

### *5.1. Convergence and Consistency*

#### *5.1.1. m-consistency, mesh-imprint effects*

The *linear (1-) consistency* as a constraint required to achieve a desired level of accuracy in gradient calculations of finite-volume methods was discussed by Barth and Jesperson in [5]. This is a highly desired property of the reconstruction, which is not only the measure of accuracy and robustness, but also a feature which minimizes mesh imprint effects.

It is well known that the least-squares reconstruction of the in-cell gradients (i.e., our $\mathrm{rDG}_{\mathrm{P_0P_1}}$) is linearly-consistent [5]. This means that linear field is reconstructed *exactly* (to round-off), on any mesh. Since our rDG is hierarchical, all $\mathrm{rDG}_{\mathrm{P_nP_{n+1}}}$ for $n > 0$ are linearly consistent as well, as verified in [44]. The method however is not quadratically consistent on *irregular* meshes. This is because the second derivatives in reference space are different from those in physical space (due to transformation), and since the method is hierarchical in reference space, there is no guarantee that a quadratic shape in physical space is reconstructed exactly. However, if the reconstructed function is quadratic in reference space, it is reconstructed exactly, as demonstrated using a regular mesh[16] in [44]. Similar, the $\mathrm{rDG}_{\mathrm{P_2P_3}}$

[16] On regular meshes, the reference and physical space are consistent, in terms of spatial derivatives.

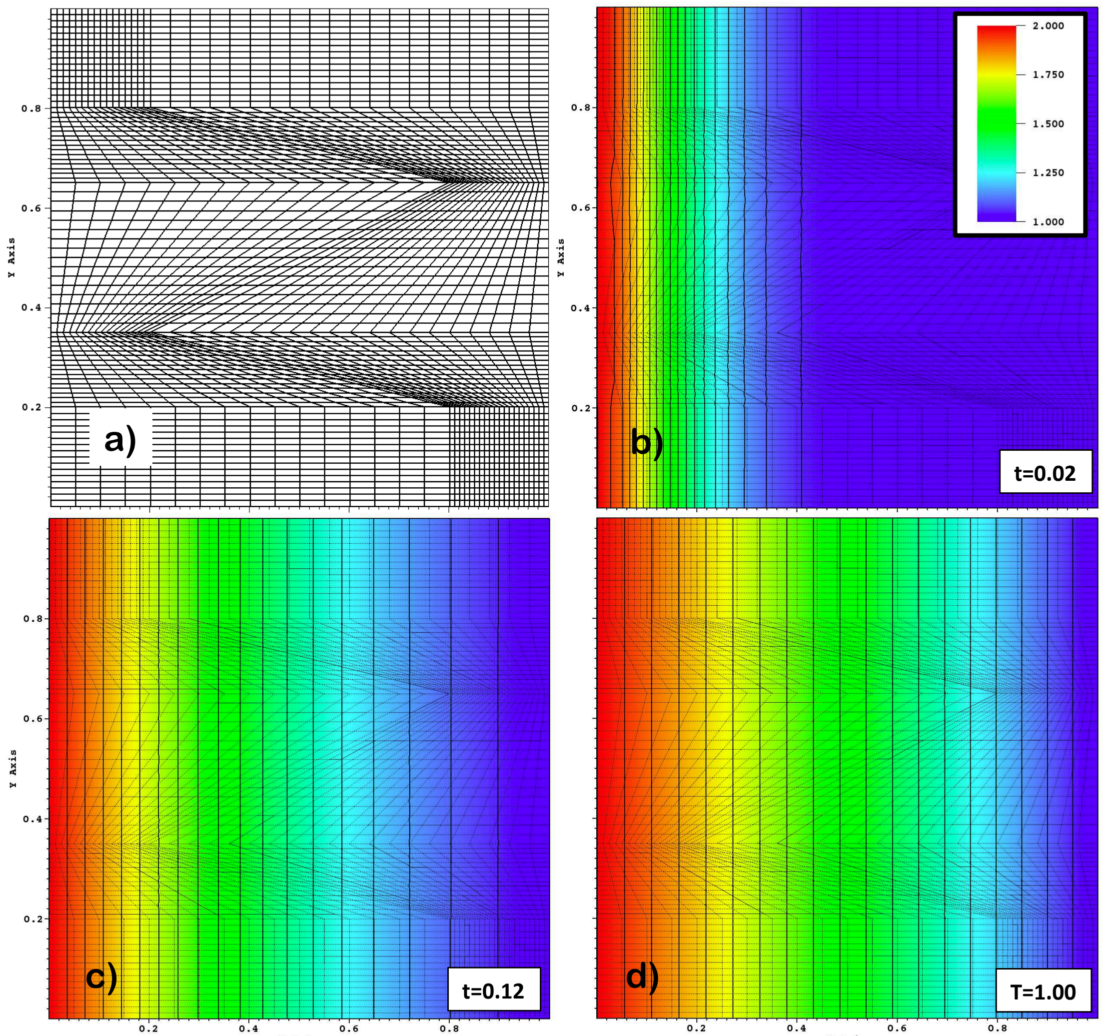


Figure 4: Kershaw heat diffusion test, using the $\mathrm{rDG}_{\mathbb{P}_2\mathbb{P}_3}$ .

method is cubically consistent in reference space. While both the $\mathrm{rDG}_{\mathrm{P}_0\mathrm{P}_1}$ and $\mathrm{rDG}_{\mathrm{P}_1\mathrm{P}_2}$ are not cubically consistent, the reconstruction errors diminish with mesh refinement, with the expected second- and the third-order, for the $\mathrm{rDG}_{\mathrm{P}_0\mathrm{P}_1}$ and the $\mathrm{rDG}_{\mathrm{P}_1\mathrm{P}_2}$ scheme, respectively (see [44] for demonstration).

To demonstrate the method's ability to compute a diffusion operator through a mesh with highly skewed elements, we utilize the problem introduced by Kershaw in [25]. In the computational domain of size $1\times1$, we generated a mesh with a distinct Z pattern, Figure 4(a). We solve the non-linear heat conduction equation in temperature formulation (see [44]), with temperature-dependent thermal conductivity, $\kappa(T) = \kappa_0 + \kappa_1 T$, where $\kappa_0 = 1$ and $\kappa_1 = \frac{1}{10}$. Initial temperature is set to 1. Neumann boundary conditions are applied at horizontal walls. For vertical walls, we use Dirichlet boundary conditions, with $T_{\mathrm{LFT}} = 2$ at the left wall, and $T_{\mathrm{RGT}} = 1$ at the right wall. Computations are performed with the 3$^{\mathrm{rd}}$-order accurate ESDIRK$_3$ time discretization, using time step $\Delta t = 10^{-2}$, which corresponds to Fourier number $Fo = \frac{\alpha\Delta t}{h^2} \sim 2000$, where $h$ is the scale of the smallest elements and $\alpha$ is the coefficient of thermal diffusivity. Nearly steady-state is achieved at $t = 1$.

Computational results are shown in Figure 4(b,c,d), for the 4$^{\mathrm{th}}$ order-accurate space discretization $\mathrm{rDG}_{\mathrm{P}_2\mathrm{P}_3}$. It can be clearly seen that no mesh-imprint "chevron" patterns are observed, resulting in nearly perfect one-dimensional temperature field, with vertical straight isolines of temperature. The undesirable "chevron" pattern effects are attributed to many cell-centered schemes for diffusion operators, as reported by Rebourcet in [52]. We believe that negligible mesh-imprinting of our scheme is attributed to the above-discussed $m$-consistency.

### 5.1.2. *Manufactured solution*

Next, we demonstrate the accuracy and convergence of the method, using the following manufactured solution for compressible Navier-Stokes equations in two dimensions:

$$\begin{array}{rcl} T(x,y,t) & = & \bar{T} + (\delta T_0 + a_T \sin(2\pi t))\cos(2\pi(x+\mathrm{v}_1 t))\sin(2\pi(y+\mathrm{v}_2 t)) \\ P(x,y,t) & = & \bar{P} + (\delta P_0 + a_P \sin(2\pi t))\sin(2\pi(x+\mathrm{v}_1 t))\cos(2\pi(y+\mathrm{v}_2 t)) \\ v_1(x,y) & = & (\delta V_0 + a_v \sin(2\pi t))\cos(2\pi(x+\mathrm{v}_1 t))\sin(2\pi(y+\mathrm{v}_2 t)) \\ v_2(x,y) & = & (\delta V_0 + a_v \sin(2\pi t))\sin(2\pi(x+\mathrm{v}_1 t))\cos(2\pi(y+\mathrm{v}_2 t)) \end{array} \tag{58}$$

where $\bar{T}$, $\bar{P}$, $\delta T_0$, $\delta P_0$, $\delta V_0$, $a_T$, $a_P$, $a_v$, $\mathrm{v}_1$, $\mathrm{v}_2$ are given constants.

The solution eq.(58) corresponds to translating (with velocity $\mathbf{w} = (\mathrm{v}_1, \mathrm{v}_2)$) and oscillating (with amplitudes $a_T$, $a_P$ and $a_v$) waves. In the following simulations, we set $\mathbf{w} = \left(\frac{1}{10}, \frac{1}{10}\right)$, $\bar{P} = 1$, $\bar{T} = 1$, $\delta P_0 = \frac{1}{10}$, $\delta T_0 = \frac{1}{10}$, $\delta V_0 = 10^{-4}$, $a_P = \frac{1}{20}$, $a_v = \frac{1}{100}$, and $a_T = \frac{1}{20}$. We used the $\gamma$-gas equation of state (Appendix A), with $\gamma = 1.4$. Both thermal conductivity and dynamic viscosity are set to be constant, $\kappa = \frac{1}{10}$ and $\mu = \frac{1}{10}$. Source terms generating this manufactured solution are computed using the symbolic manipulation in *Mathematica*. We used the computational domain shown in Figure 5. There are six geometrical parameters, describing the computational domain: $X_0 = -1$, $X_1 = 1$, $Y_0 = -\frac{1}{2}$, $Y_1 = \frac{1}{2}$, $\alpha = 10^{\circ}$ and $\beta = 10^{\circ}$. The $\alpha$ and $\beta$ are generally varied, to allow different levels of mesh stretching/distortion. Dirichlet boundary conditions are applied at all boundaries of the domain.

First, we measure space convergence rates, by using the 5$^{\mathrm{th}}$-order-accurate time discretization, ESDIRK$_5$, and setting time stepping to $\left(t = 2\times\Delta t = 10^{-3}\right)$. This ensures that time discretization errors are smaller than space discretization errors. The results are shown in Figure 6, for the convergence of the $\mathcal{L}_2$-norm velocity-magnitude errors. As one can see, all three rDG schemes do converge consistently – i.e., with the second order for the $\mathrm{rDG}_{\mathrm{P}_0\mathrm{P}_1}$, with the third-order for the $\mathrm{rDG}_{\mathrm{P}_1\mathrm{P}_2}$, and with (approximately) the fourth-order for the $\mathrm{rDG}_{\mathrm{P}_2\mathrm{P}_3}$.

Time convergence is demonstrated in Figure 7, for the $\mathcal{L}_2$-norm of the velocity magnitude error. In the simulations, we used the $\mathrm{rDG}_{\mathrm{P}_2\mathrm{P}_3}$ on a mesh with 32,762 elements, to ensure small spatial discretization errors. This space resolution is sufficient to measure nearly asymptotic convergence rates for time discretization schemes up to the 3$^{\mathrm{rd}}$-order accurate. The 4$^{\mathrm{th}}$- and 5$^{\mathrm{th}}$-order accurate ESDIRK$_{4,5}$ schemes exhibit nearly the 4$^{\mathrm{th}}$-order convergence rate when time steps are large. The convergence flattens for smaller time steps, when space discretization errors become dominant.

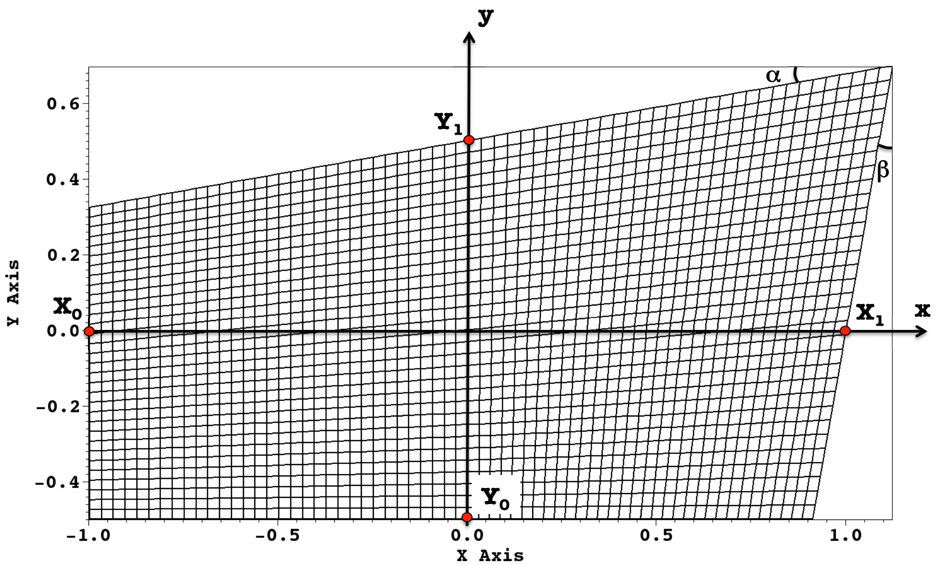


Figure 5: Domain and mesh setup for the manufactured-solution test problem. $\alpha = \beta = 10^{\circ}$.

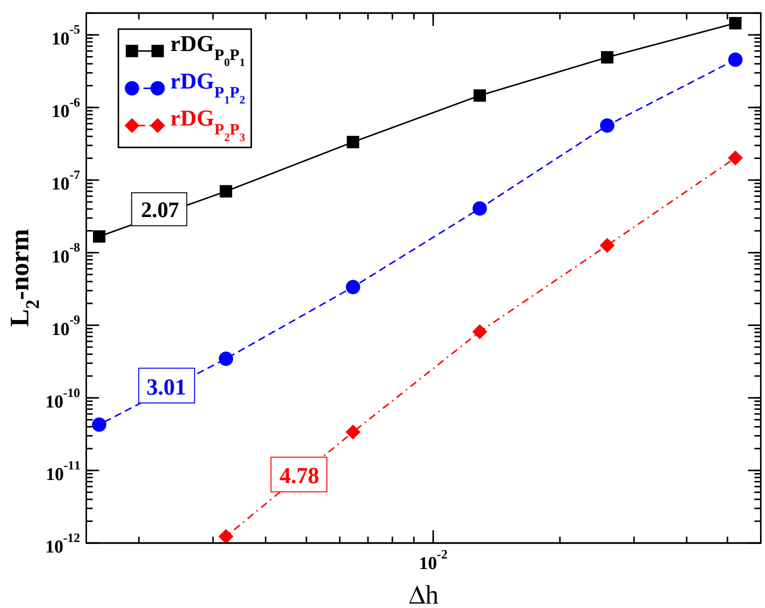


Figure 6: Convergence of the velocity magnitude field, with mesh refinement and different space discretization schemes.

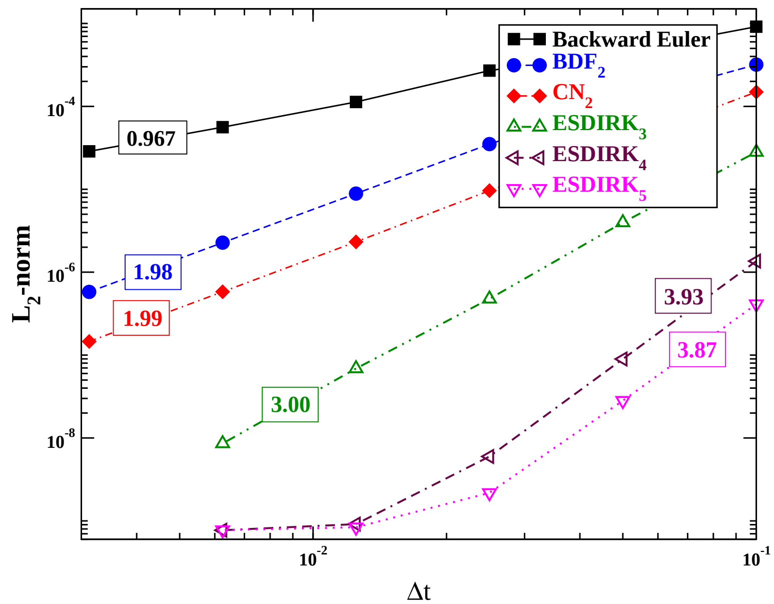


Figure 7: Convergence of the velocity magnitude field, with time step refinement and different time discretization schemes.

## *5.2. Forced Convection*

### *5.2.1. Driven-cavity thermal flows*

Next, we move to forced-convection fluid flow in an enclosed domain. The "Lid-Driven Cavity" (LDC) problem has been established as a standard "benchmark" test for numerical methods of (nearly-) incompressible fluid dynamics. In [18], Ghia et al. employed a finite-difference method, using a steady-state vorticity-stream-function ($\omega - \psi$) formulation of incompressible flow, to obtain solutions in a wide range of Reynolds ($Re$) numbers. Using grids with high resolution (129×129 and 257×257) and "coupled strongly-implicit multigrid" method, Ghia et al. provided "reference" data for comparison of velocity profiles at the vertical and horizontal centerlines of the square cavity, with no-slip vertical and bottom walls, and a moving top wall. Here, we will simulate this problem using the compressible (thermal) formulation, setting the flow Mach number ($M$) to very low values. We used the 2-parameter EOS, as described in Appendix A. Most of the simulations shown here are done with $M = 10^{-2}$. For the energy equation, we set isothermal boundary conditions at vertical walls ($T_{\mathrm{LFT}} = 1$ and $T_{\mathrm{RGT}} = 2$), and adiabatic boundary conditions at horizontal walls.

First, we verified the code for the low-Reynolds case of $Re = 400$, which corresponds to a steady-state flow field. Solution results are compared to those of [18], and presented in [44] – all showing an excellent agreement. Our 4$^{\mathrm{th}}$-order rDG$_{\mathrm{P_2P_3}}$ captures the secondary vortices in the lower corners of the cavity (both – the BR$_1$ and BL$_1$, following notation from [18]), with a grid resolution as low as 32×32. Moreover, the centerline velocity profiles are indistinguishable from those reported in [18], even for grid resolution as small as 16×16, as compared to the reference solution by Ghia et al. on the mesh of 129×129.

Here, we will be focusing on the high-$Re$-number case of $10^4$, when the flow dynamics is more rich, and the advantages of high-order schemes become more evident. Snapshots of the flow evolution are shown in Figures 8 and 9. As expected for these high $Re$ numbers, the solution is unsteady, and the eddies are constantly evolving: forming, breaking, disappearing[17]. We observe numerous secondary vortices formed, BR$_{1,2}$ and BL$_{1,2}$, which are dynamically

[17] In the present study, we are using LLNL's visualization tool VisIt, with recently added "multi-resolution" option to render high-order piecewise-discontinuous solution representations in elements. In essence, VisIt allows to refine original elements, and high-order in-cell solution is mapped

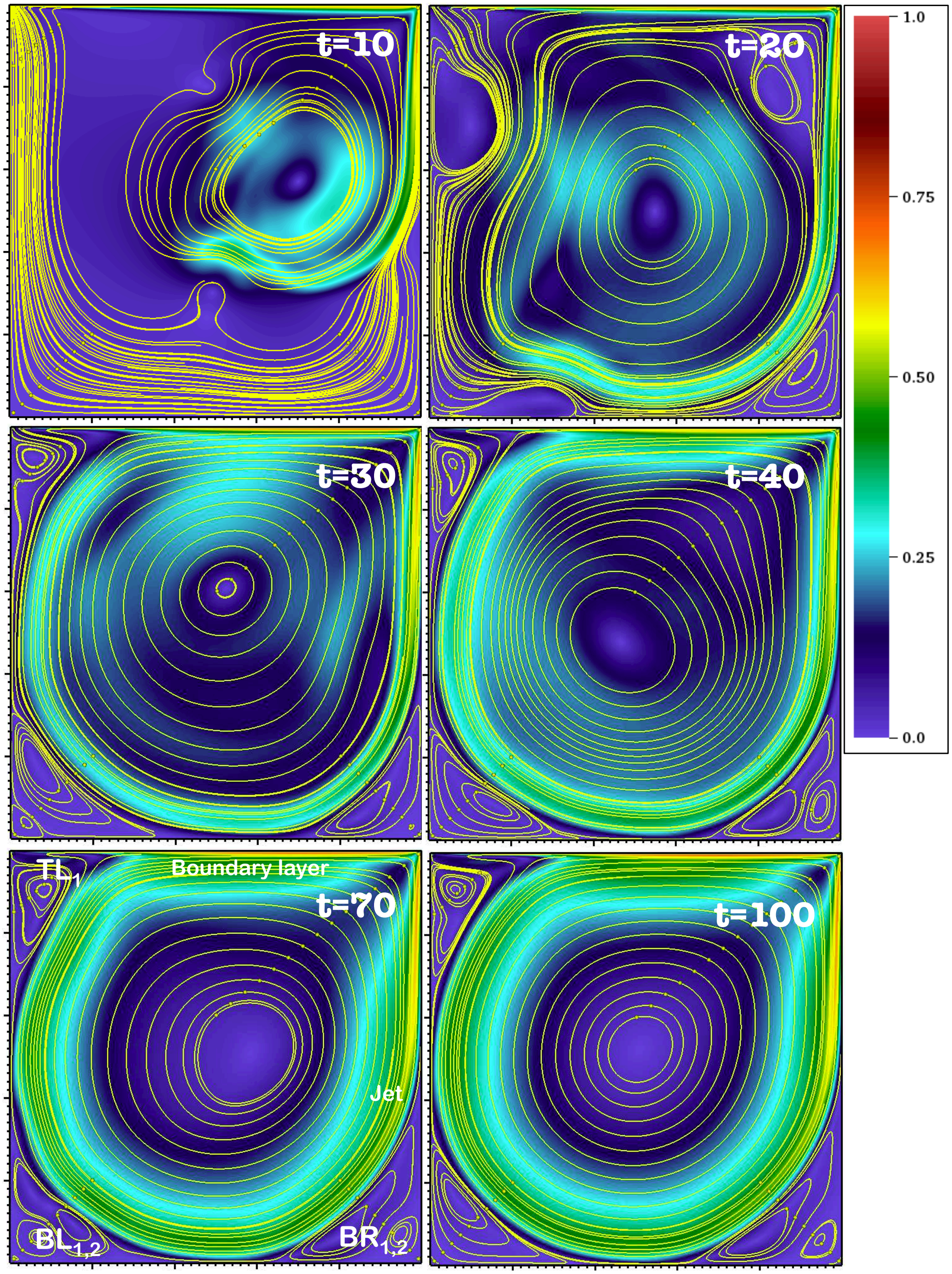


Figure 8: Dynamics of the velocity magnitude and streamlines for $rDG_{P_2P_3}$, mesh resolution 128×128, ESDIRK$_5$, and $\Delta t = 1$. $Re = 10^4$.

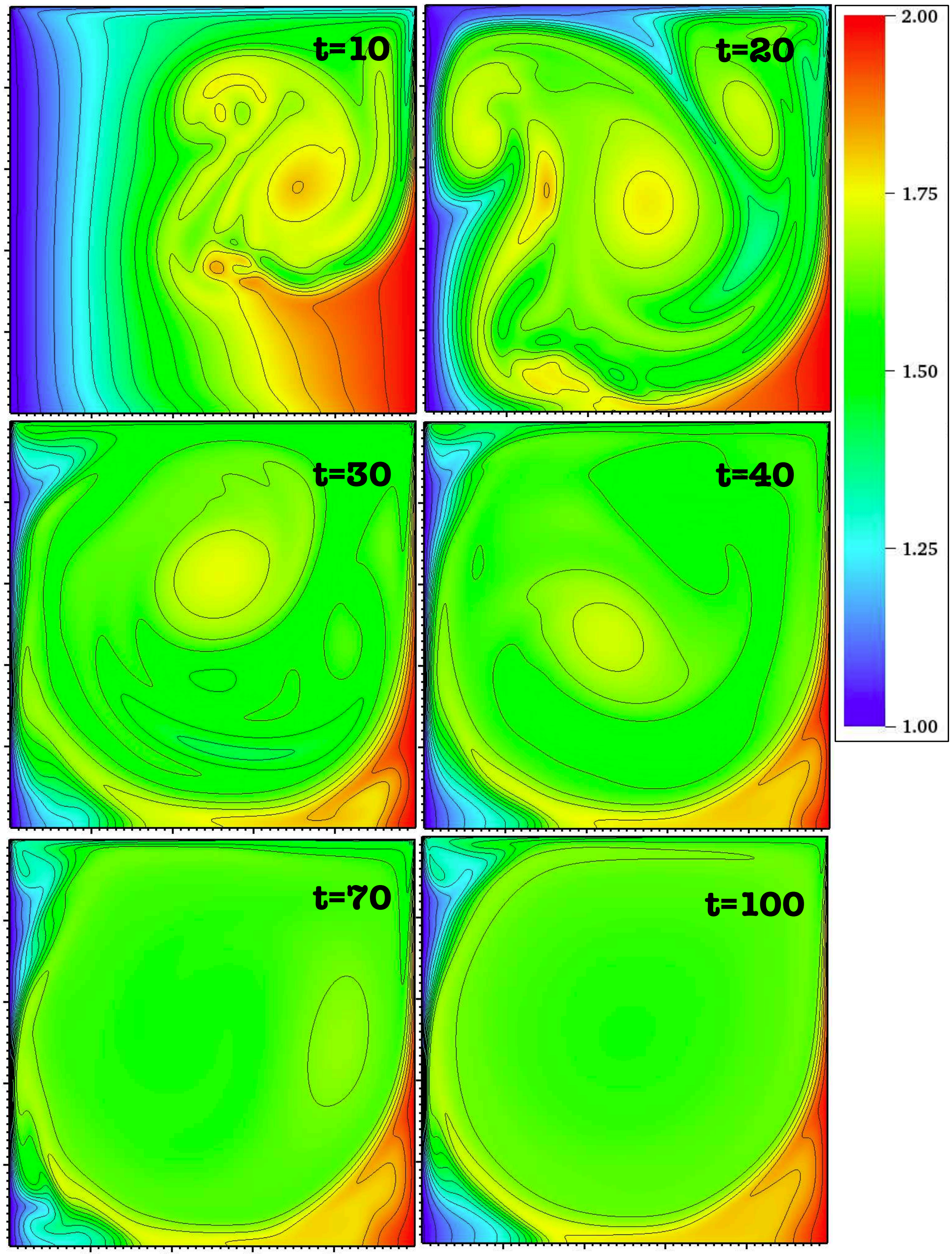


Figure 9: Dynamics of the temperature field for $\mathrm{rDG}_{P_2P_3}$, mesh resolution 128×128, $\mathrm{ESDIRK}_5$, and $\Delta t = 1$. $Re = 10^4$.

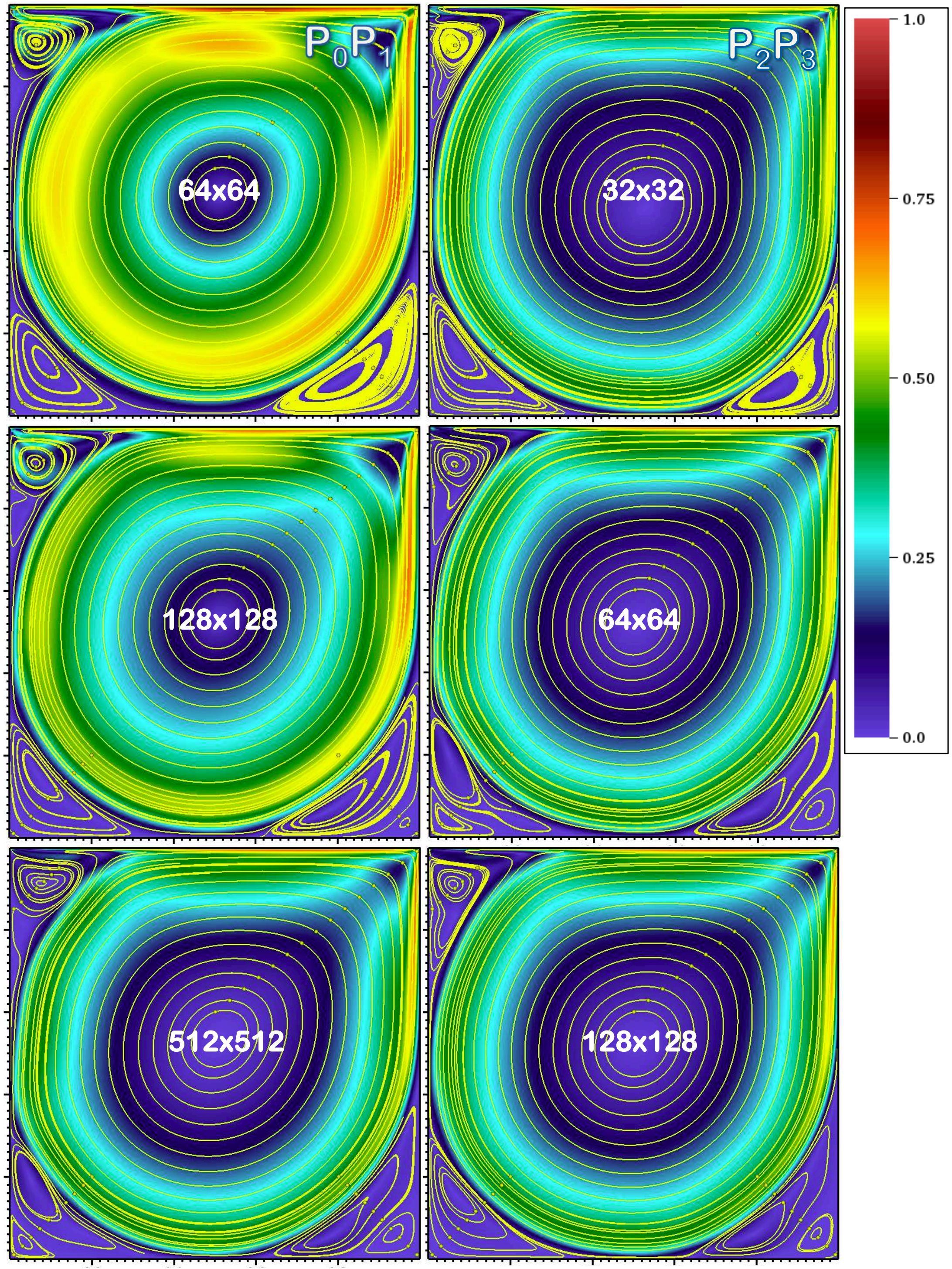


Figure 10: On mesh convergence of the velocity magnitude and streamlines, rDG$_{P_0P_1}$ (left) vs. rDG$_{P_2P_3}$ (right). ESDIRK$_5$ and $\Delta t$ = 1. $Re$ = $10^4$. $Pr$ = 0.7.

growing and decaying[18]. Also, the large central vortex is unstable and gyrating with a period of approximately 10 time units. The jet is also unsteady, wiggling and causing the vortical flows in the corners of the cavity to become unstable, providing enhanced thermal mixing in the domain. It is instructive to note that we ran these simulations with large time steps ($\mathrm{CFL}_{\mathrm{mat}}$=128), without significant loss of accuracy. This is because of the 5th-order-accurate *L*-stable time discretization scheme $\mathrm{ESDIRK}_5$. Similar simulations using the 2nd-order $\mathrm{BDF}_2$, with the same time step, exhibit significant dissipation, damping instabilities of the flow field. To get comparable (i.e., no noticeable damping) simulations with $\mathrm{BDF}_2$, one needs to run with $\mathrm{CFL}_{\mathrm{mat}} < 32$. It is important to note that the material[19] CFL does not define the dynamic time scales of this problem. Rather, the *dynamic time scale* corresponds to the *lifetime of secondary eddies*. Therefore, one could (and should) run these simulations with time steps significantly exceeding the residence time of particles in elements. This can be achieved with fully-implicit time discretization schemes[20], like deployed here, provided that time discretization errors are negligible.

Table 1: $\mathrm{rDG}_{\mathrm{P_0P_1}}$ vs. $\mathrm{rDG}_{\mathrm{P_2P_3}}$

| **Measure** | $\mathrm{rDG}_{\mathrm{P_0P_1}}$ $(512 \times 512)$ | $\mathrm{rDG}_{\mathrm{P_2P_3}}$ $(64 \times 64)$ | **Benefit** |
|---|---|---|---|
| # Elements | 262,144 | 4,096 | $\frac{1}{64}$ |
| # DoFs | 1,048,576 | 96,304 | $\frac{1}{11}$ |
| $\mathrm{CFL}_{\mathrm{mat}} = \frac{\lvert\mathbf{v}\rvert\Delta t}{h}$ | 512 | 64 | $\frac{1}{8}$ |
| $\mathrm{CFL}_{\mathrm{aco}} = \frac{c\Delta t}{h}$ | 51,200 | 6,400 | $\frac{1}{8}$ |
| $\mathrm{Fo}_{\mu} = \frac{\mu\Delta t}{\rho h^2}$ | 26 | 0.41 | $\frac{1}{64}$ |

An example of the mesh convergence is demonstrated in Figure 10. As one can see, with the 4th-order-accurate $\mathrm{rDG}_{\mathrm{P_2P_3}}$, an adequate flow resolution is achieved with the 64×64 mesh[21]. The 2nd-order-accurate $\mathrm{rDG}_{\mathrm{P_0P_1}}$ is excessively diffusive, damping instabilities and smearing ("thickening") both the boundary layer and the jet. Statistically comparable-to-$\mathrm{rDG}_{\mathrm{P_2P_3}}$ (and adequate) $\mathrm{rDG}_{\mathrm{P_0P_1}}$ solution is achieved with a mesh resolution of 512×512. The benefits of using high-order scheme are outlined in Table 5.2.1. In conjunction with the underlying linear algebra, we would like to emphasize that the high-order method involves an order-of-magnitude smaller solution vector, which is generally better for approximate matrix inversion algorithms. In addition, all important CFL and Fo numbers are significantly lower, due to larger mesh sizes, which is very important for better conditioning of linear steps in the Newton iteration algorithm. However, these factors have to be put together with a significantly higher cost per DoF

into the embedded refined mesh, with the user-defined refinement ratio, achieving high-resolution solution images even on very coarse meshes (see [44] for details).

[18] This high-*Re* case was also presented in [18]. However, Ghia et al. used a steady-state formulation, missing important dynamical flow features.

[19] Because of the compressible formulation, we use two distinct CFL numbers – one based on material velocity, $\mathrm{CFL}_{\mathrm{mat}}$, and another based on sound speed, $\mathrm{CFL}_{\mathrm{aco}}$.

[20] It is also instructive to note that operator-splitting algorithms, like variations of the *projection* method [6, 9, 19], are inaccurate when deployed with $\mathrm{CFL}_{\mathrm{mat}} \gg 1$. Moreover, when $\mathrm{CFL}_{\mathrm{mat}} \gg 10$, these methods become unstable [42].

[21] In fact, the general flow features are captured well even on a coarse, 32×32, mesh, with only some small-scale vortex structures missed due to very coarse mesh in the lower corners, grossly under-resolving secondary eddies. Moreover, the gyrating motion of the central vortex is captured very well, in difference to the 2nd-order-accurate $\mathrm{rDG}_{\mathrm{P_0P_1}}$, which damps this motion completely, for mesh resolutions below 128×128.

for the $\mathrm{rDG}_{\mathrm{P_2P_3}}$. Our estimate of the performance (in terms of CPU time) for our implementation of the algorithm shows that the $\mathrm{rDG}_{\mathrm{P_2P_3}}$ on $64 \times 64$ mesh is roughly three times faster than the same simulation with the $\mathrm{rDG}_{\mathrm{P_0P_1}}$ on a $512 \times 512$ mesh. We believe that the benefit of the high-order schemes can be significantly higher, if one could exploit a potential for enhancing the algorithms performance, using recent disruptive trends in the HPC node architecture, such as GPUs and RAJA portability layer [24]. We believe that some features of the high-order rDG, such as locality and the relatively high cost per DoF (compared to the low-order variants) would make the rDG highly compatible with node-level parallel programming models and multi-threading.

*5.2.2. Shedding Flow past a Triangular Wedge*

In our next example, we demonstrate performance of our 4th-order $\mathrm{rDG}_{\mathrm{P_2P_3}}$, for capturing unstable vortical flow in the external flow past an obstacle. We placed a two-dimensional wedge (an equilateral triangle, with the side length 1) in a computational domain of size 40×30, five units downstream of the inlet. An inflow boundary condition is applied at the left, outflow (Neumann) boundary conditions are enforced at the right, and freestream boundary conditions are set at the top and bottom. At the wedge surface, no-slip isothermal boundary conditions are enforced. We used the $\gamma$-gas EOS ($\gamma = 1.4$). The Reynolds number (based on the wedge-side length and freestream velocity) is set to $Re = 10^2$, while the Prandtl number was set to $Pr = 0.7$. Simulations are performed for freestream $M_\infty = 10^{-2}$. More details of the computational setup are given in [44].

In Figure 11, we show solutions for the velocity magnitude field, when the unsteady Kármán street is fully established, comparing the $\mathrm{rDG}_{\mathrm{P_0P_1}}$ and the $\mathrm{rDG}_{\mathrm{P_2P_3}}$, on a sequence of successively refined meshes. The coarsest mesh of 2,055 elements is also shown in Figure 11(e). Notice that the mesh has high-aspect-ratio elements in the boundary layer. Also, there are transition zones with high ratio of element sizes. We intentionally use this (generally considered "bad") mesh – as many discretization schemes are known to struggle on highly distorted, stretched, high-aspect-ratio meshes.

With the 4th-order $\mathrm{rDG}_{\mathrm{P_2P_3}}$, the vortex shedding is captured on the coarsest mesh, Figure 11(a,e). We would like to emphasize that because of the high-order piecewise-cubic in-cell solution representation, the size of the captured vortices are comparable to the element size. The piecewise-linear $\mathrm{rDG}_{\mathrm{P_0P_1}}$ kills these flow features on this mesh, resulting in a steady wake behind the wedge, even for the finer mesh of 8,220 elements. With a further refinement (of 32,880 elements), the 2nd-order $\mathrm{rDG}_{\mathrm{P_0P_1}}$ is able to capture the Kármán street, as well, but at a cost of 16 times more elements and > 2.5 times more total DoFs solved-for.

*5.3. Natural circulation*

*5.3.1. Thermally-driven flow*

In this example, we add buoyancy. We will skip verification tests, and refer to [44], where we performed an extensive benchmarking for thermally-driven flows in square cavities (comparing to [12]), and for Rayleigh-Bénard convection (comparing to [41]). Here, we show the results for natural circulation in a non-square domain, as defined by Figure 5, using $\alpha = \beta = 20^\circ$. The flow is driven by gravity, due to the temperature difference $\Delta\hat{T} = \hat{T}_{\mathrm{LFT}} - \hat{T}_{\mathrm{RGT}}$, applied at the left and right walls. The top and bottom walls are adiabatic. We use the 2-parameter EOS, see Appendix A. Simulations are performed on a sequence of Rayleigh numbers $Ra = 10^4$, $10^5$ and $10^6$. The Prandtl number was fixed at $Pr = 0.71$. The maximum Mach number was varied from $10^{-2}$ to $10^{-4}$, depending on the $Ra$ number. For buoyancy, we utilized the *Boussinesq* approximation. Computations are done using the $\mathrm{BDF}_2$ time discretization. For all solved equations, non-linear tolerances are set to $10^{-7}$. For all runs, steady-state solutions have been achieved, running with time steps corresponding to material CFL numbers greater than 50.

Computational results are shown in Figure 12, for three $Ra$ numbers. As one can see, with the increase in $Ra$ number, the boundary layers become thinner, and the overall vortex flow pattern becomes more complex, forming a secondary eddy, for $Ra = 10^6$. All flow features are captured well with the 4th-order $\mathrm{rDG}_{\mathrm{P_2P_3}}$, using the 64×32 mesh.

*5.3.2. Natural convection with solid crust formation*

In the next example, we modified the previous test, by adding solidification at the left wall. Initially, the fluid is motionless, and the temperature is set well above the liquidus. In dimensionless units, $\hat{T}_{\mathrm{init}} = 2$. The liquidus and

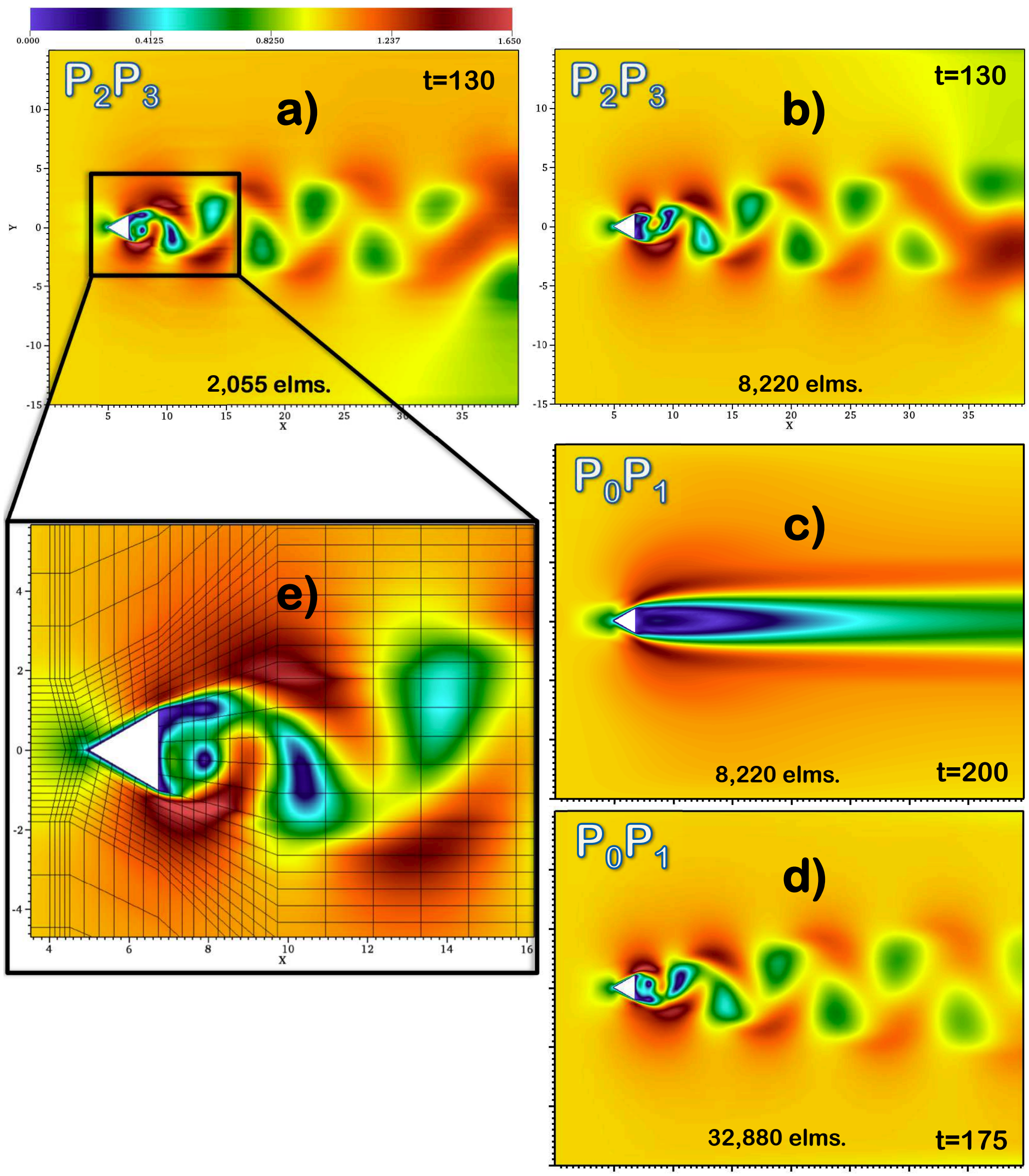


Figure 11: Capturing vortex shedding behind a triangular wedge. Magnitude of the velocity vector, for quasi-steady solution, using $\mathrm{rDG}_{P_2P_3}$ vs. $\mathrm{rDG}_{P_0P_1}$.

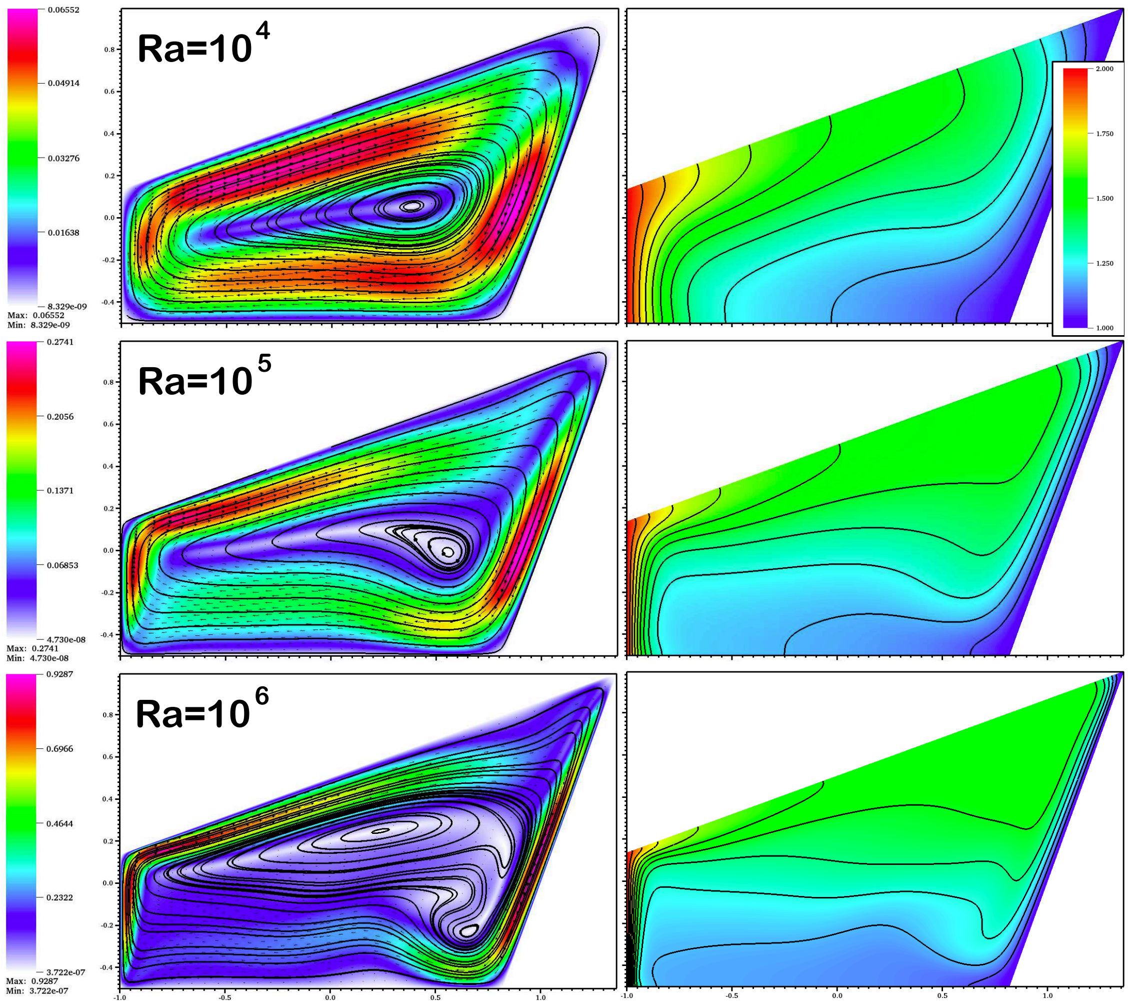


Figure 12: Velocity/streamline (left) and temperature (right) distributions for different *Ra* numbers, with the rDG$_{P_2P_3}$ on the mesh 64×32.

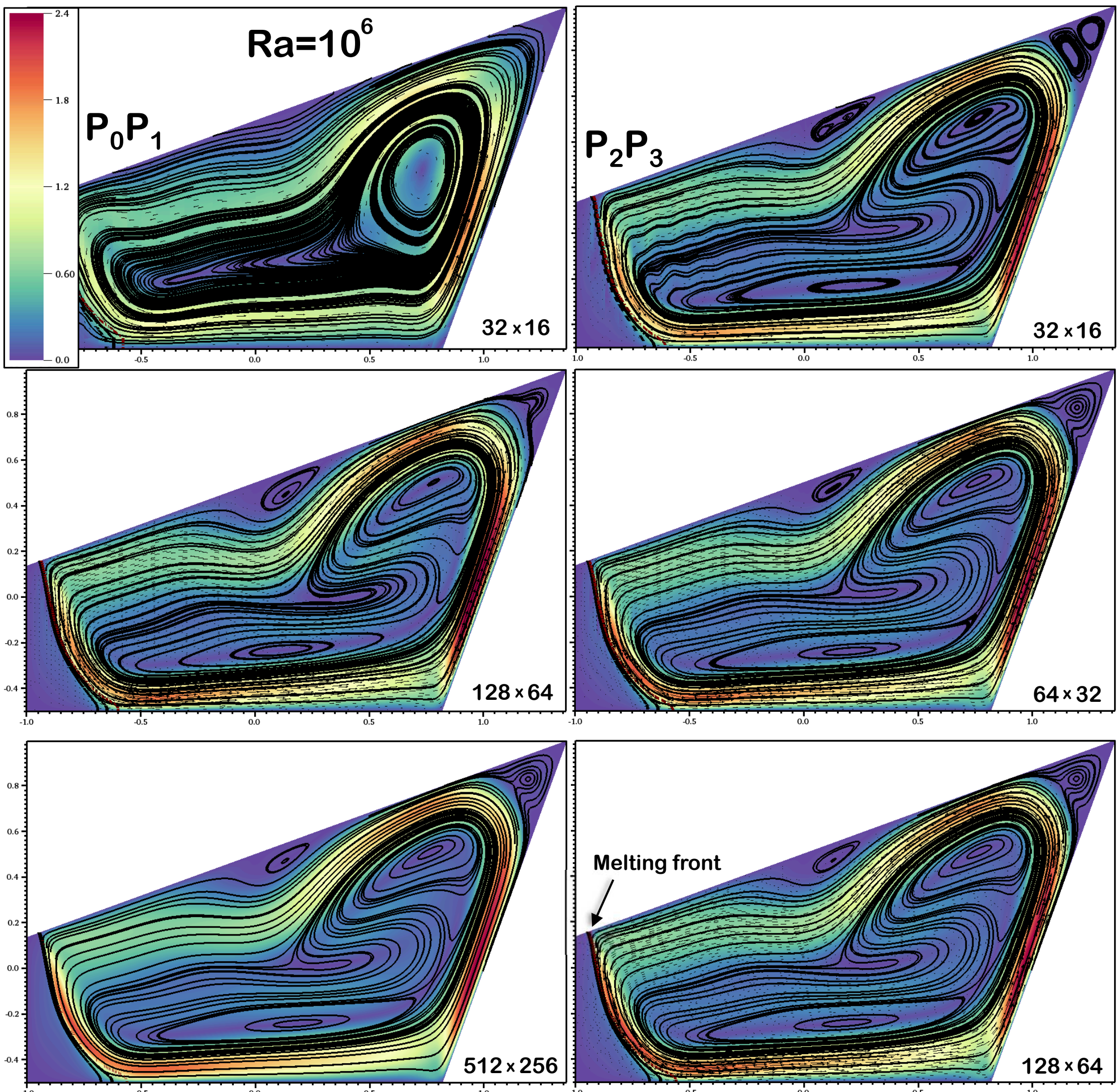


Figure 13: Mesh convergence for velocity/streamline fields. $\text{rDG}_{\text{P}_0\text{P}_1}$ vs. $\text{rDG}_{\text{P}_2\text{P}_3}$. $Ra = 10^6$.

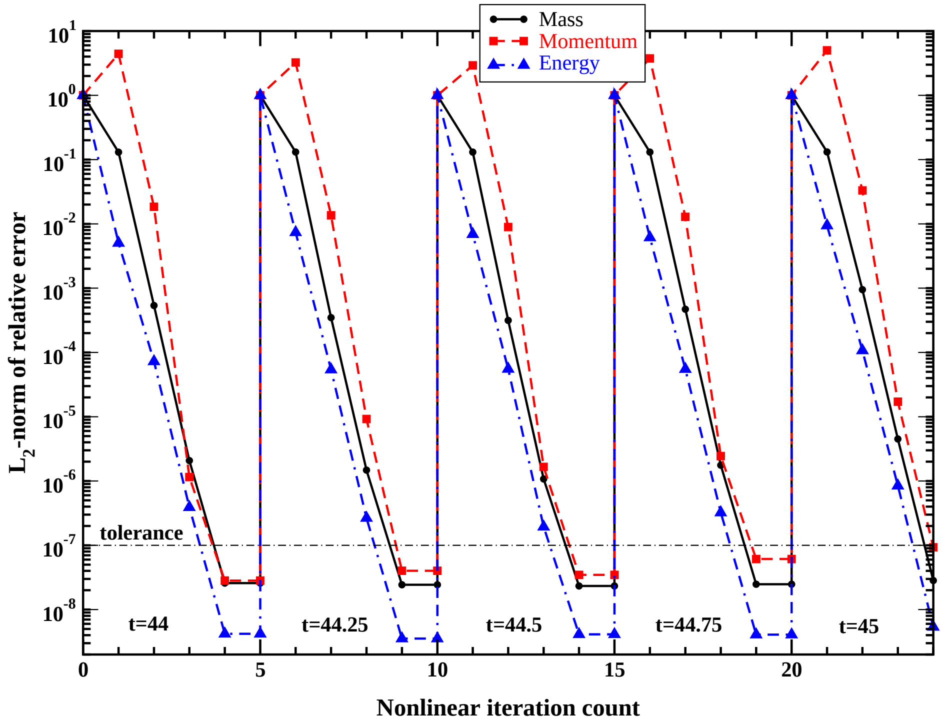


Figure 14: History of the non-linear convergence for five time steps of the simulation with $\mathrm{rDG}_{\mathrm{P}_2\mathrm{P}_3}$ on mesh 128×64 and $Ra = 10^6$. $\mathrm{CFL}_{\mathrm{aco}} = 7,550$, $\mathrm{CFL}_{\mathrm{mat}} = 5.31$, $\mathrm{Fo}_\kappa = 8$, $\mathrm{Fo}_\mu = 800$.

solidus temperatures were defined to be $\hat{T}_L = 1.5$ and $\hat{T}_S = 1.4$, respectively. The transients are started dropping the left wall temperature down to the liquidus temperature (smoothly, within one dimensionless time unit), and keeping it at this level for awhile so that a nearly steady-state natural circulation is established, and then dropping the left wall temperature more, below the solidus, down to $\hat{T}_{\mathrm{LFT}} = 1$. The right wall temperature was kept const, at $\hat{T}_{\mathrm{RGT}} = 2$, well above the melting point. The top and bottom walls are kept adiabatic. This initiated a formation of the solid crust layer, Figure 13. We used the viscosity-based material strength model, as discussed in Section 2.2 and Appendix B. The parameters of this model are $f_S = 10^2$, $\alpha = 10$ and $\omega = 2$, which corresponds to the variation of the viscosity factor as depicted in Figure 19. The scaling parameters for this test are: $Ra = 10^6$, $Gr = 10^7$, $Re = 3,162$, $Pr = 0.1$ and $Ste = 4.854$.

Simulations are performed with the $2^{\mathrm{nd}}$-order $\mathrm{BDF}_2$ time discretization, setting the non-linear tolerances at the level of $10^{-7}$. A sample of the non-linear convergence is shown in Figure 14. The convergence curves are typical for quadratic rates (slightly curved upwards), at the first 2-3 iterations, before leveling off, at the asymptotic limit, when the round-off errors become dominant.

The mesh convergence for the $\mathrm{rDG}_{\mathrm{P}_0\mathrm{P}_1}$ is demonstrated in Figure 13 (left). It can be seen that the $2^{\mathrm{nd}}$-order discretization errors tend to suppress resolution of small vortical structures. Only with the mesh as large as $512 \times 256$, we are getting the full resolution of all five vortical structures. Comparing to the $\mathrm{rDG}_{\mathrm{P}_2\mathrm{P}_3}$ (Figure 13, right), this requires an order of magnitude more degrees of freedom, which effectively corresponds to significantly larger linear algebra matrices to invert, and significantly larger CFL/Fo numbers involved (therefore, the stiffer the underlying linear solver). In terms of the CPU time, we do not get ten-fold speed-up, because the $\mathrm{rDG}_{\mathrm{P}_2\mathrm{P}_3}$ is more expensive per DoF, than the $\mathrm{rDG}_{\mathrm{P}_0\mathrm{P}_1}$ – but still achieve a better (2-3 times) performance.

As noted in Section 2.2, phase transition is extremely difficult to treat numerically. This is due to very tight

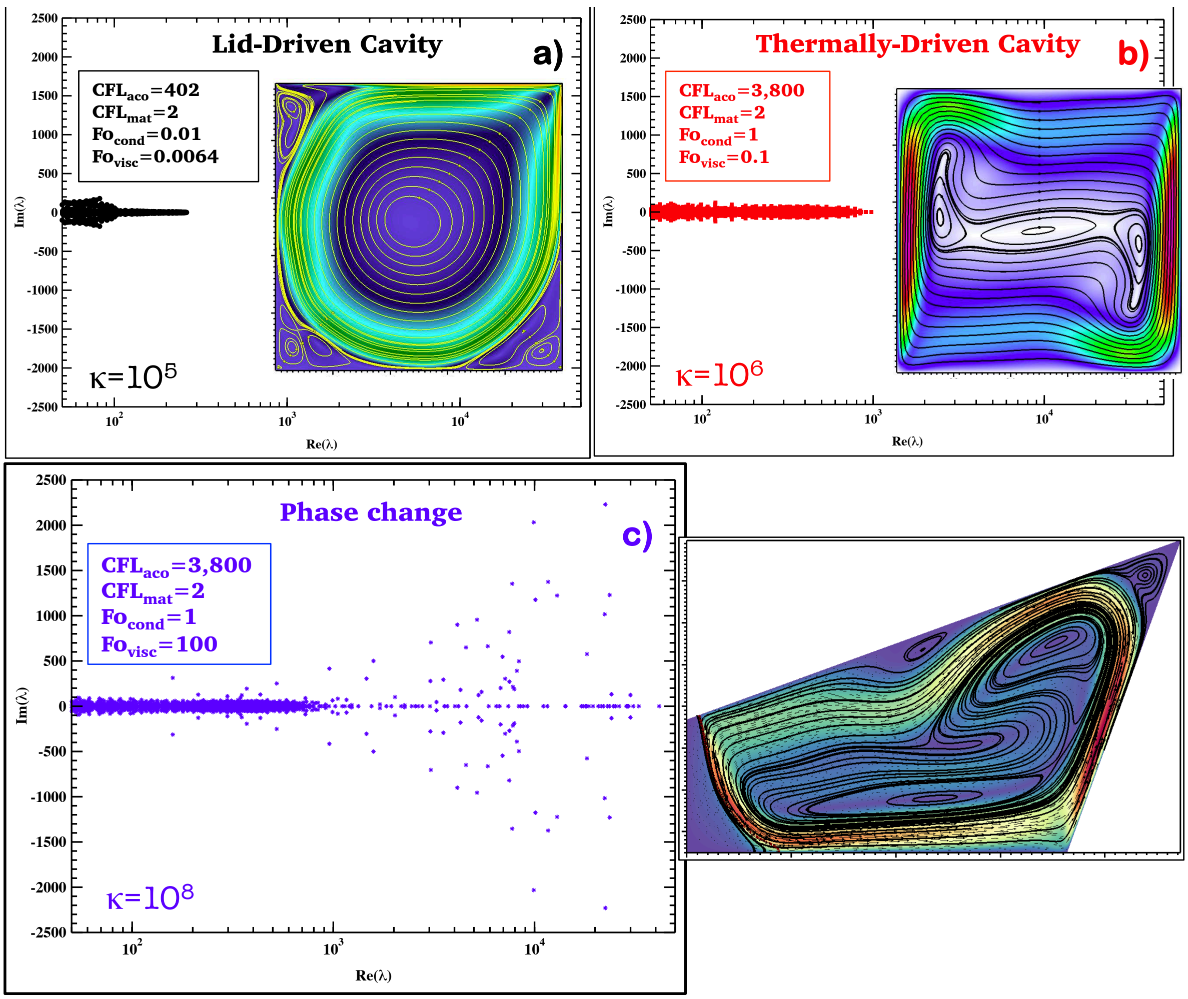


Figure 15: On "eigen-scopy" of the Jacobian matrices for different problems considered in the present study.

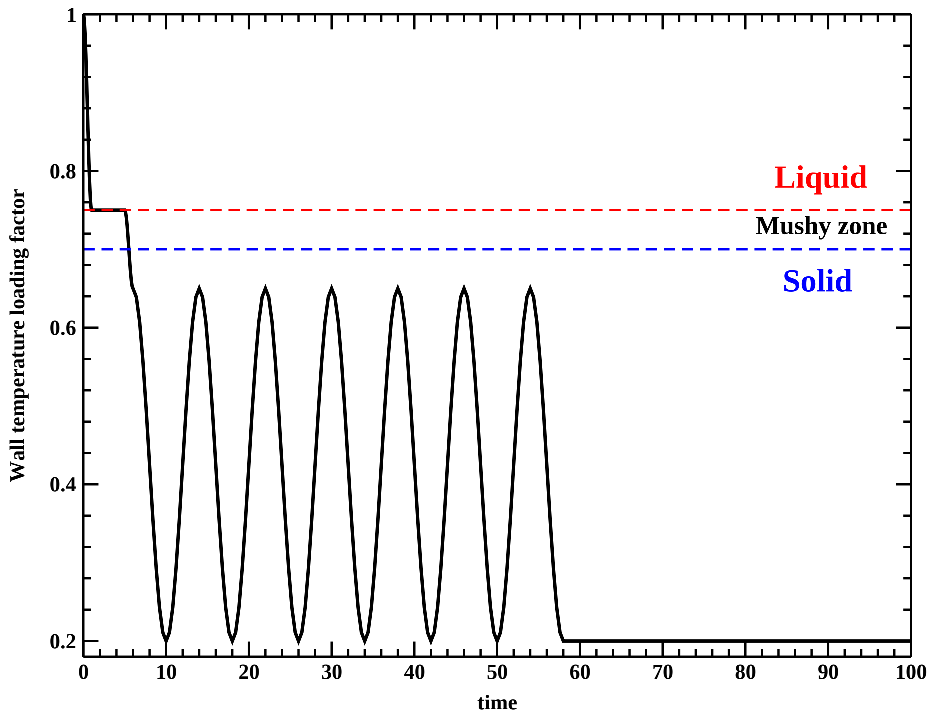


Figure 16: Thermal loading curve for the melt pool convection test. The nominal (initial) wall temperature was $\hat{T}_{\mathrm{LFT}} = \hat{T}_{\mathrm{RGT}} = 2$.

coupling of the momentum and energy equations, which necessitates the use of the fully-implicit method, as described in Section 4. This can be seen from the eigenvalue distributions, Figure 15, as computed using the Elemental package [50], for Jacobian matrices of three numerical problems considered here. The first problem is the LDC (Section 5.2.1), corresponding to a passive transport of the energy field, with one-way weak coupling of the momentum and energy equations. The fastest time scale corresponds to the stiff acoustic (pressure) waves, which are embedded into our formulation. The condition number[22], as defined by the ratio of the largest-to-smallest eigenvalue magnitude, $\kappa$ (an indicator of the stiffness), is on the order of $10^5$, Figure 15(a). The buoyancy adds another level of complexity, by introducing a two-way coupling of the momentum and energy, through the Boussinesq source terms. The condition number for the thermally-driven flow is on the order of $\kappa = 10^6$, Figure 15(b). The phase change model introduces additional tight coupling through the variable-coefficient parabolic operator[23], due to the viscosity-based material strength model. Thus, the computed eigenvalue distribution is highly dispersed, Figure 15(c), with the condition number on the of order of a billion. Our past experience with using explicit hydro schemes [26] for this configuration exposed severe stability issues. In [41], we used a SIMPLE-based Picard-iteration non-linear solver, for similar phase transition problems in nuclear reactor severe accident applications. It took hundreds of Picard iterations to get marginally converged solution, with a Darcy-law-based phase change model [57]. In contrast, with our Newton-based approach, we are reliably getting convergence to a rather tight tolerance, in just a few (3 to 5) non-linear iterations.

### 5.4. Melt pool dynamics

In our final numerical example, we demonstrate the method's performance for melt pool convection with a solid crust formation under stable and unstable stratification. The computational domain is defined by Figure 5 ($\alpha = \beta = 20^{\circ}$), with two blocks added at the left and right, to enable a smooth boundary condition transition between

[22]Computed using the SLEPc package [23].
[23]Note, the diffusivity coefficient is varied by three orders of magnitude, over 2-3 element-thick "mushy" zone.

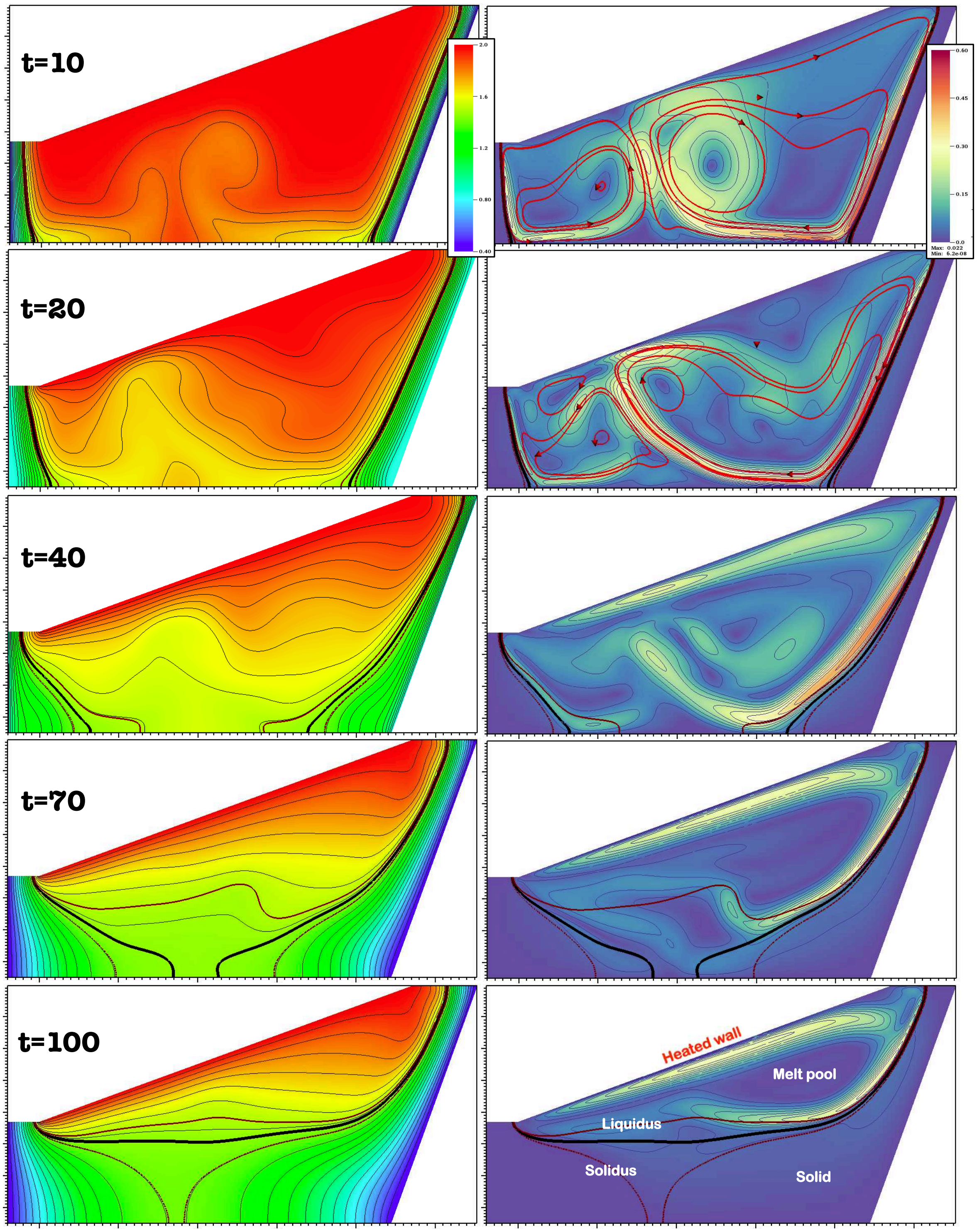


Figure 17: Melt pool dynamics with stable stratification. Temperature field (left) and velocity magnitude/streamline fields (right).

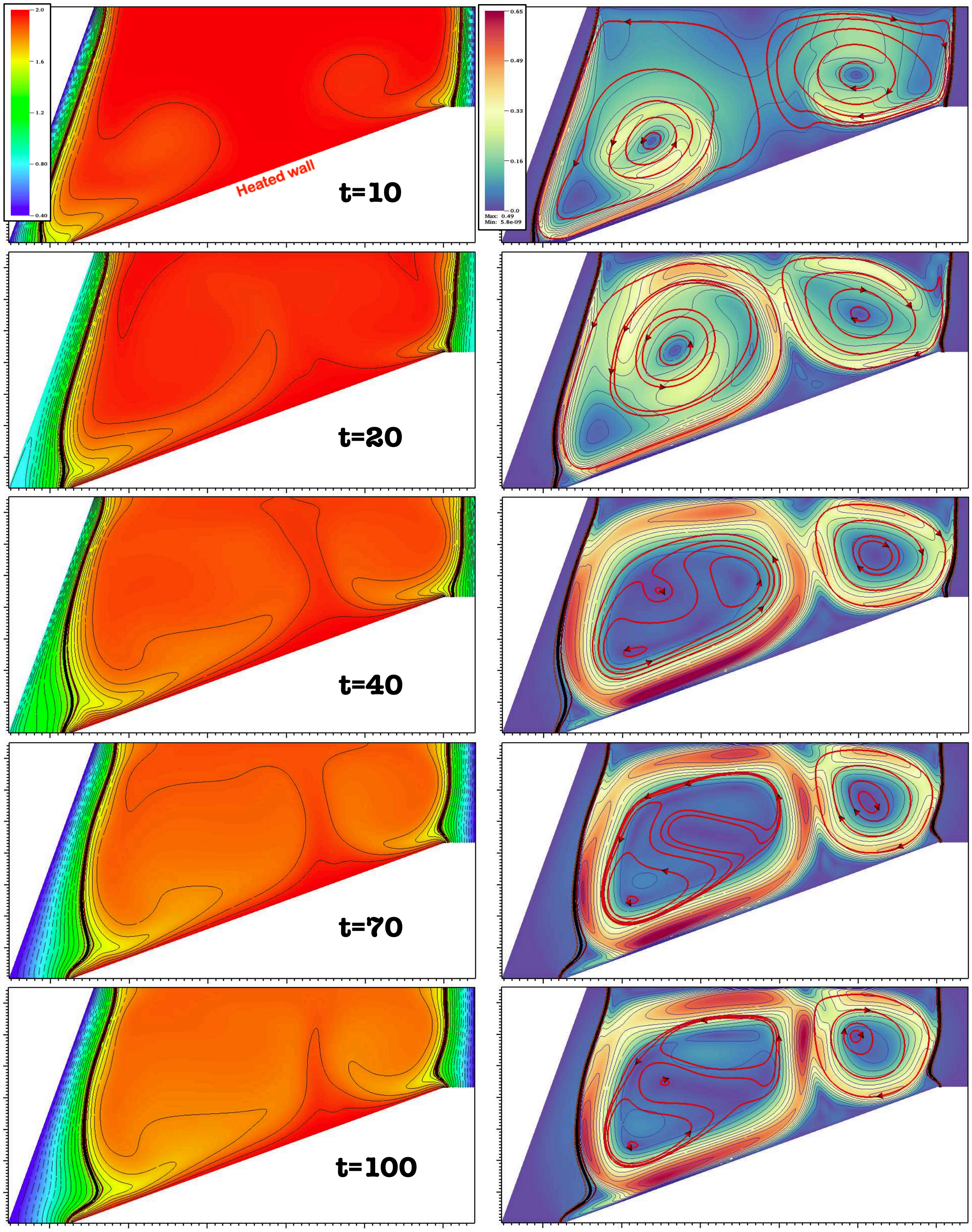


Figure 18: Melt pool dynamics with unstable stratification. Temperature field (left) and velocity magnitude/streamline fields (right).

the heated and the cooled walls. In the first configuration, the top wall is heated (the temperature is set to $\hat{T}_{\text{TOP}} = 2$). The horizontal sections of the top wall, as well as the bottom wall are kept adiabatic. The left and right walls are cooled, by applying the wall temperature loading as shown in Figure 16. To get crust evolution dynamics with solidification and re-melting, the side-wall temperatures were oscillatory, for a period of the dimensionless time from 10 to 58. The second configuration was the same, but with the reversed direction of the gravity vector. This makes an interesting case of the unstable stratification. The initial conditions were motionless (molten) fluid with a constant temperature $\hat{T}_{\text{init}} = 2$. The scaling parameters for these tests are: $Ra_{\text{init}} = 10^6$, $Pr = 0.1$ and $Ste = 6$, $M_{\text{max}} = 0.005$. All simulations are done using the 2$^{\text{nd}}$-order BDF$_2$ time discretization scheme, running with time steps $\Delta t = \frac{1}{32}$, on the 128×64 mesh, and using the 4$^{\text{th}}$-order rDG$_{\text{P}_2\text{P}_3}$ scheme. This corresponds to CFL$_{\text{mat}}$=1.6, CFL$_{\text{aco}}$=316, Fo$_{\mu}$=100 and Fo$_{\kappa}$=1.

Computational results for the first case of the heating-from-the-top (stable stratification) are shown in Figure 17. The flow dynamics starts with a formation of the natural convection, followed by the solidifying and re-melting solid crust at the side walls of the molten pool, and, finally, freezing of the most part of the domain, due to establishment of the stable stratification, damping the natural convection and inhibiting the effective heat transfer.

In the second configuration of the heating-from-the-bottom, the stratification is unstable, Figure 18. This promotes an effective mixing, inhibiting a formation of the solid crust at the bottom of the molten pool. Notably, because of the higher heat transfer, the boundary layers are thin, leading to the relatively thin solid crusts at the sides, and very thin “mushy” regions, between the solidus and liquidus.

## 6. Concluding Remarks

In this study, we introduced a new reconstructed Discontinuous Galerkin method based on *orthogonal basis/test functions* and *least-squares reconstruction*. The method is generally a subclass of the methods of Mean Weighted Residuals (MWR), in which we solve for degrees of freedom in a piecewise-polynomial *primitive-variable solution representation*, minimizing *conservation-variable residuals*, satisfying the underlying governing equations. There are two main technical contributions of the work - *a)* the in-cell and inter-cell reconstruction, using the tensor-product Legendre polynomials as basis functions, combined with inverse-Jacobian-weighted test functions, ensuring orthogonality of the mass matrix and better solvability of the fully-implicit solver; and *b)* the ability to solve for primitive variables, chosen on the basis of better conditioning/solvability of the underlying governing equations, while enforcing conservation laws. These features enable a robust combination of the rDG with the Newton-Krylov based fully-implicit solution procedure.

We have demonstrated that the newly developed method is capable of producing highly accurate solutions of difficult multiphysics problems on highly distorted/stretched unstructured grids. The scope of complexity involves multiple-time-scale problems, with a wide spread in scales, including very fast (stiff) modes. In particular, we have shown all-speed flow capabilities, when fully-compressible simulations are performed for extremely small Mach numbers $< 10^{-3}$, without explicit filtering of acoustic modes on the problem formulation (governing equations) level. Also, we have demonstrated an ability to incorporate stiff material strength models within the framework of the multi-material systems with fluid-solid phase change (melting/solidification).

Our future effort will concentrate on three areas. First, on *improvement of preconditioning*. In the present study, we evaluate approximate (lagged) Jacobian matrices, and utilize LU factorizations, as preconditioning techniques for the GMRES based linear steps in the Newton iterations. This is a rather expensive approach, in both the memory requirements (for the matrix storage), and the CPU time (for the LU factorizations involved). Instead, we are developing and implementing a preconditioning strategy which involves iterative procedures, based on equation-splitting combined with *p*-multigrid. Our preliminary experimentation indicates that this is a promising strategy to achieve a scalable performance. Second, we need to extend our current modeling approach to work with *multi-material interfaces*. Our current plan in this direction is to involve a combination of the level set and volume tracking algorithms to represent interfaces between (partially molten) powder material and ambient gas. Third, we will explore the proposed

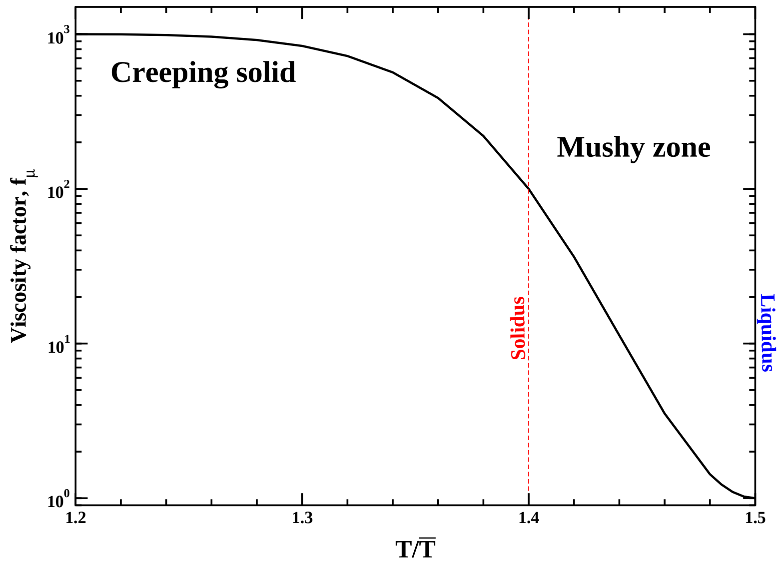


Figure 19: An example of the viscosity factor, as a function of temperature. $\hat{T}_L = 1.5$, $\hat{T}_S = 1.4$, $f_S = 10^2$, $\alpha = 10$ and $\omega = 2$.

methodology for phase change with boiling/condensation, as needed in the selective laser melting (SLM) of our interest here, and potentially impactful in many other applications, such as cavitating flows [16] and numerous challenging nuclear reactor safety problems [43, 61].

**Acknowledgments**

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344, and funded by the Laboratory Directed Research and Development Program at LLNL under project tracking code 13-SI-002.

## A. 7-parameter equation of state

The equation of state is defined as

$$P(\rho, \mathfrak{u}) = \mathcal{A}_0 + \mathcal{A}_1\eta + \mathcal{A}_2\eta^2 + \mathcal{A}_3\eta^3 + \left(\mathcal{B}_0 + \mathcal{B}_1\eta + \mathcal{B}_2\eta^2\right)\mathfrak{u} \tag{59}$$

where

$$\eta(\rho) = \frac{\rho}{\rho_0} - 1 \tag{60}$$

and $\mathcal{A}_{0,1,2,3}$, $\mathcal{B}_{0,1,2}$ and $\rho_0$ are given constants.

Setting $\mathcal{A}_j = 0$, $\mathcal{B}_2 = 0$ and $\mathcal{B}_0 = \mathcal{B}_1 = \rho_0(\gamma - 1)$, we will get the *$\gamma$-law gas*. Another useful limiting case is $\mathcal{A}_{2,3} = \mathcal{B}_j = 0$, which is denoted as the *2-parameter EOS*. In this case, the speed of sound is constant, and defined as $c = \sqrt{\frac{\mathcal{A}_1}{\rho_0}}$.

## B. Solid-state viscosity model

In the present study, the viscosity factor in eq.(11) is defined as

$$f_\mu(T) = 10^{\varphi(T)} \tag{61}$$

where

$$\begin{aligned} \varphi(T) &= -(a_0 - 4a_1)\,\psi(T) + 2(a_0 - 2a_1)\,\psi(T)^2 \\ \psi(T) &= \tfrac{1}{2}\left(1 + \cos\left(\pi\left(b_0 + b_1\hat{T} + b_2\hat{T}^2\right)\right)\right) \end{aligned} \tag{62}$$

$$\begin{aligned} & a_0 = \log_{10}(\alpha f_S), \quad a_1 = \log_{10}(f_S) \\ b_0 &= \frac{\hat{T}^\star\left(\hat{T}_L\left(\hat{T}^\star - \hat{T}_L\right) - 2\hat{T}_S\left(\hat{T}^\star - \hat{T}_S\right)\right)}{d} \\ b_1 &= \frac{\hat{T}^{\star 2} + \hat{T}_L^2 - 2\hat{T}_S^2}{d} \\ b_2 &= -\frac{\hat{T}^\star + \hat{T}_L - 2\hat{T}_S}{d} \\ d &= 2\left(\hat{T}^\star - \hat{T}_L\right)\left(\hat{T}^\star - \hat{T}_S\right)\left(\hat{T}_L - \hat{T}_S\right) \\ \hat{T}^\star &= \hat{T}_L - \omega\left(\hat{T}_L - \hat{T}_S\right) \end{aligned} \tag{63}$$

and $\hat{T} = \frac{T}{T}$ is the dimensionless temperature. There are three input parameters for this model, i.e. $- f_S$ (viscosity factor at solidus), $\alpha$ (defining the limiting "solid-state" viscosity as $\mu_S^\star = \alpha f_S \mu_L$) and $\omega$ (defining the thickness of the "creeping solid" state). An example of the $f_\mu(T)$ is shown in Figure 19.

## C. Mapping DoFs

### C.1. Change of basis functions, Legendre-to-Taylor

For evaluation of diffusion operators and for reconstruction/recovery, it is necessary to have a mapping from the orthogonal basis functions eq.(16) to the Taylor basis functions eq.(15). This mapping is provided by matrices denoted as $\mathbb{L}_{2T}^{(P_n)}$. Defining the vector of the orthogonal Legendre-based DoFs in an element as $\mathbf{U}_{(k)}$, and the vector of the Taylor-based DoFs as $\mathbf{V}_{(k)}$, the third-order mapping $\mathbb{L}_{2T}^{(P_2)}$ in 2D is

$$\underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & \xi_{x_0} & \eta_{x_0} & 0 & 0 & 0 \\ 0 & \xi_{y_0} & \eta_{y_0} & 0 & 0 & 0 \\ 0 & \Lambda^{(P_2)}_{2T_{2,4}} & \Lambda^{(P_2)}_{2T_{3,4}} & 3\xi_{x_0}^2 & 2\xi_{x_0}\eta_{x_0} & 3\eta_{x_0}^2 \\ 0 & \Lambda^{(P_2)}_{2T_{2,5}} & \Lambda^{(P_2)}_{2T_{3,5}} & 3\xi_{x_0}\xi_{y_0} & \xi_{x_0}\eta_{y_0} + \xi_{y_0}\eta_{x_0} & 3\eta_{x_0}\eta_{y_0} \\ 0 & \Lambda^{(P_2)}_{2T_{2,6}} & \Lambda^{(P_2)}_{2T_{3,6}} & 3\xi_{y_0}^2 & 2\xi_{y_0}\eta_{y_0} & 3\eta_{y_0}^2 \end{bmatrix}}_{\mathbb{L}_{2T}^{(P_2)}} \begin{bmatrix} \mathbf{U}_{(0)} \\ \mathbf{U}_{(1)} \\ \mathbf{U}_{(2)} \\ \mathbf{U}_{(3)} \\ \mathbf{U}_{(4)} \\ \mathbf{U}_{(5)} \end{bmatrix} = \begin{bmatrix} \mathbf{V}_{(0)} \\ \mathbf{V}_{(1)} \\ \mathbf{V}_{(2)} \\ \mathbf{V}_{(3)} \\ \mathbf{V}_{(4)} \\ \mathbf{V}_{(5)} \end{bmatrix} \tag{64}$$

where

$$\begin{aligned} \Lambda^{(P_2)}_{2T_{2,4}} &= \eta_{x_0}\partial_\eta\xi_{x_0} + \xi_{x_0}\partial_\xi\xi_{x_0} & \Lambda^{(P_2)}_{2T_{3,4}} &= \eta_{x_0}\partial_\eta\eta_{x_0} + \xi_{x_0}\partial_\xi\eta_{x_0} \\ \Lambda^{(P_2)}_{2T_{2,4}} &= \eta_{y_0}\partial_\eta\xi_{x_0} + \xi_{y_0}\partial_\xi\xi_{x_0} & \Lambda^{(P_2)}_{2T_{3,4}} &= \eta_{y_0}\partial_\eta\eta_{x_0} + \xi_{y_0}\partial_\xi\eta_{x_0} \\ \Lambda^{(P_2)}_{2T_{2,4}} &= \eta_{y_0}\partial_\eta\xi_{y_0} + \xi_{y_0}\partial_\xi\xi_{y_0} & \Lambda^{(P_2)}_{2T_{3,4}} &= \eta_{y_0}\partial_\eta\eta_{y_0} + \xi_{y_0}\partial_\xi\eta_{y_0} \end{aligned} \tag{65}$$

and the coefficients of the inverse-Jacobian matrix, and their derivatives are evaluated at element's centers. This mapping is developed on the requirement that both the Legendre-based and the Taylor-based solution representations in an element give identical point-wise solutions and all derivatives (up to the order of interest), at the geometrical center of the element.

Similarly, the fourth-order mapping is defined as

$$\mathbb{L}^{(P_3)}_{2T}\begin{bmatrix}\mathbf{U}_{(0)}\\ \mathbf{U}_{(1)}\\ \dots\\ \mathbf{U}_{(9)}\end{bmatrix}=\begin{bmatrix}\mathbf{V}_{(0)}\\ \mathbf{V}_{(1)}\\ \dots\\ \mathbf{V}_{(9)}\end{bmatrix} \tag{66}$$

$$\mathbb{L}^{(P_3)}_{2T}=\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & \xi_{x_0} & \eta_{x_0} & 0 & 0 & 0 & \Lambda^{(P_3)}_{2T_{7,2}} & \Lambda^{(P_3)}_{2T_{8,2}} & \Lambda^{(P_3)}_{2T_{9,2}} & \Lambda^{(P_3)}_{2T_{10,2}}\\
0 & \xi_{y_0} & \eta_{y_0} & 0 & 0 & 0 & \Lambda^{(P_3)}_{2T_{7,3}} & \Lambda^{(P_3)}_{2T_{8,3}} & \Lambda^{(P_3)}_{2T_{9,3}} & \Lambda^{(P_3)}_{2T_{10,3}}\\
0 & \Lambda^{(P_3)}_{2T_{2,4}} & \Lambda^{(P_3)}_{2T_{3,4}} & \Lambda^{(P_3)}_{2T_{4,4}} & \Lambda^{(P_3)}_{2T_{5,4}} & \Lambda^{(P_3)}_{2T_{6,4}} & \Lambda^{(P_3)}_{2T_{7,4}} & \Lambda^{(P_3)}_{2T_{8,4}} & \Lambda^{(P_3)}_{2T_{9,4}} & \Lambda^{(P_3)}_{2T_{10,4}}\\
0 & \Lambda^{(P_3)}_{2T_{2,5}} & \Lambda^{(P_3)}_{2T_{3,5}} & \Lambda^{(P_3)}_{2T_{4,5}} & \Lambda^{(P_3)}_{2T_{5,5}} & \Lambda^{(P_3)}_{2T_{6,5}} & \Lambda^{(P_3)}_{2T_{7,5}} & \Lambda^{(P_3)}_{2T_{8,5}} & \Lambda^{(P_3)}_{2T_{9,5}} & \Lambda^{(P_3)}_{2T_{10,5}}\\
0 & \Lambda^{(P_3)}_{2T_{2,6}} & \Lambda^{(P_3)}_{2T_{3,6}} & \Lambda^{(P_3)}_{2T_{4,6}} & \Lambda^{(P_3)}_{2T_{5,6}} & \Lambda^{(P_3)}_{2T_{6,6}} & \Lambda^{(P_3)}_{2T_{7,6}} & \Lambda^{(P_3)}_{2T_{8,6}} & \Lambda^{(P_3)}_{2T_{9,6}} & \Lambda^{(P_3)}_{2T_{10,6}}\\
0 & \Lambda^{(P_3)}_{2T_{2,7}} & \Lambda^{(P_3)}_{2T_{3,7}} & \Lambda^{(P_3)}_{2T_{4,7}} & \Lambda^{(P_3)}_{2T_{5,7}} & \Lambda^{(P_3)}_{2T_{6,7}} & \Lambda^{(P_3)}_{2T_{7,7}} & \Lambda^{(P_3)}_{2T_{8,7}} & \Lambda^{(P_3)}_{2T_{9,7}} & \Lambda^{(P_3)}_{2T_{10,7}}\\
0 & \Lambda^{(P_3)}_{2T_{2,8}} & \Lambda^{(P_3)}_{2T_{3,8}} & \Lambda^{(P_3)}_{2T_{4,8}} & \Lambda^{(P_3)}_{2T_{5,8}} & \Lambda^{(P_3)}_{2T_{6,8}} & \Lambda^{(P_3)}_{2T_{7,8}} & \Lambda^{(P_3)}_{2T_{8,8}} & \Lambda^{(P_3)}_{2T_{9,8}} & \Lambda^{(P_3)}_{2T_{10,8}}\\
0 & \Lambda^{(P_3)}_{2T_{2,9}} & \Lambda^{(P_3)}_{2T_{3,9}} & \Lambda^{(P_3)}_{2T_{4,9}} & \Lambda^{(P_3)}_{2T_{5,9}} & \Lambda^{(P_3)}_{2T_{6,9}} & \Lambda^{(P_3)}_{2T_{7,9}} & \Lambda^{(P_3)}_{2T_{8,9}} & \Lambda^{(P_3)}_{2T_{9,9}} & \Lambda^{(P_3)}_{2T_{10,9}}\\
0 & \Lambda^{(P_3)}_{2T_{2,10}} & \Lambda^{(P_3)}_{2T_{3,10}} & \Lambda^{(P_3)}_{2T_{4,10}} & \Lambda^{(P_3)}_{2T_{5,10}} & \Lambda^{(P_3)}_{2T_{6,10}} & \Lambda^{(P_3)}_{2T_{7,10}} & \Lambda^{(P_3)}_{2T_{8,10}} & \Lambda^{(P_3)}_{2T_{9,10}} & \Lambda^{(P_3)}_{2T_{10,10}}
\end{bmatrix} \tag{67}$$

where we only show a non-zero pattern, for brevity.

### *C.2. Change of formulation,* **W***-to-***U**

For evaluation of time derivatives when solving for primitive variables, we need to map DoFs defined for **W** to those of **U**. In the present work, we use a Gauss quadrature based approach, which is defined by the following relationship:

$$\begin{aligned}\mathbf{U}_{(n)} &\equiv \mathcal{A}_{(n)}\iint_{\Omega_e}\left(\sum_{k=0}^{K-1}\mathbf{W}_{(k)}\mathcal{B}_{(k)}(\xi,\eta)\right)\mathcal{W}_{(n)}d\Omega\\ &= \mathcal{A}_{(n)}\sum_{g=0}^{N_g}\left(\omega_g\mathcal{B}_{(n)}\left(\xi_g,\eta_g\right)\sum_{k=0}^{K-1}\left(\mathbf{W}_{(k)}\mathcal{B}_{(k)}\left(\xi_g,\eta_g\right)\right)\right)\end{aligned} \tag{68}$$

where $\omega_g$, and $\left(\xi_g,\eta_g\right)$ are the weight and the reference-space coordinates of the Gauss integration points, respectively, while the $\mathbf{W}_{(k)}$ are the (solved-for) DoFs in the primitive-variable **W**-formulation.

## References


[1] (2013). ALE3D Web page. https://wci.llnl.gov/simulation/computer-codes/ale3d.

[2] (2014). ALE3D users manual, An Arbitrary Lagrangian/Eulerian 2D and 3D Code System. Technical Report LLNL-SM-650174 - Version 4.22, Lawrence Livermore National Laboratory.

[3] Aris, R. (1962). *Vectors, Tensors, and the Basic Equations of Fluid Mechanics*. Dover Publications, Inc., New York.

[4] Balay, S., Abhyankar, S., Adams, M., Brown, J., Brune, P., Buschelman, K., Eijkhout, V., Gropp, W., Kaushik, D., Knepley, M., McInnes, L. C., Rupp, K., Smith, B., and Zhang, H. (2014). PETSc users manual. Technical Report ANL-95/11 - Revision 3.5, Mathematics and Computer Science Division, Argonne National Laboratory.

[5] Barth, T. and Jesperson, P. (1989). The design and application of upwind schemes on unstructured meshes. In *AISS-90-0013, AIAA 28th Aerospace Sciences Meeting*, Reno, Nevada, USA.

[6] Bell, J. B., Colella, P., and Glaz, H. M. (1989). A second-order projection method for the incompressible Navier-Stokes equations. *Journal of Computational Physics*, 85:257–283.

[7] Bijl, H., Carpenter, M., Vatsa, V., and Kennedy, C. (2002). Implicit time integration schemes for the unsteady compressible Navier-Stokes equations: Laminar flow. *Journal of Computational Physics*, 179:313–329.

[8] Carpenter, M., Kennedy, C., Bijl, H., Vilken, S., and Vatsa, V. (2005). Fourth-order Runge-Kutta schemes for fluid mechanics applications. *SIAM Journal of Scientific Computing*, 25:157–194.

[9] Chorin, A. J. (1968). Numerical solution of the Navier-Stokes equations. *Mathematics of Computations*, 22:745–762.

[10] Coleman, T. F. and Moré, J. J. (1983). Estimation of sparse Jacobian matrices and graph coloring problems. *SIAM Journal of Numerical Analysis*, 20(1):187–209.

[11] Dantzig, J. A. (1989). Modelling liquid-solid phase changes with melt convection. *International Journal for Numerical Methods in Engineering*, 28:1769–1785.

[12] de Vahl Davis, G. (1983). Natural convection of air in a square cavity: a bench mark numerical solution. *International Journal for Numerical Methods in Fluids*, 3:249–264.

[13] Dennis, J.E, J. and Schnabel, R. B. (1983). *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall, Inc., Englewood Cliffs, NJ.

[14] Dumbser, M. (2010). Arbitrary high order $P_nP_m$ schemes on unstructured meshes for the compressible Navier-Stokes equations. *Computers & Fluids*, 39:60–76.

[15] Dumbser, M., Balsara, D., Toro, E., and Munz, C. (2008). A unified framework for the construction of one-step finite volume and discontinuous Galerkin schemes on unstructured meshes. *Journal of Computational Physics*, 227:8209–8253.

[16] Dumbser, M., Iben, U., and Munz, C. (2013). Efficient implementation of high order unstructured WENO schemes for cavitating flows. *Computers & Fluids*, 86:141–168.

[17] Dumbser, M. and Zanotti, O. (2009). Very high order $P_nP_m$ schemes on unstructured meshes for the resistive relativistic MHD equations. *Journal of Computational Physics*, 228:6991–7006.

[18] Ghia, U., Ghia, K., and Shin, C. (1982). High-*Re* solutions for incompressible flow using the Navier-Stokes equations and a Multigrid method. *Journal of Computational Physics*, 48:347–411.

[19] Gresho, P. M. (1990). On the theory of semi-implicit projection methods for viscous incompressible flow and its implementation via a finite element method that also introduces a nearly consistent mass matrix. Part 1: Theory. *International Journal for Numerical Methods in Fluids*, 11:587–620.

[20] Haire, E. and Wanner, G. (1996). *Solving ordinary differential equations II: Stiff and differential-algebraic problems*. Springer-Verlag, Berlin, 2nd edition.

[21] Harlow, F. H. and Amsden, A. A. (1968). Numerical calculation of almost incompressible flow. *Journal of Computational Physics*, 3:80–93.

[22] Harlow, F. H. and Amsden, A. A. (1971). A numerical fluid dynamics calculation method for all flow speeds. *Journal of Computational Physics*, 8:197–213.

[23] Hernandez, V., Roman, J., and Vidal, V. (2005). SLEPc: A scalable and flexible toolkit for the solution of eigenvalue problem. *ACM Trans. Math. Software*, 31(3):351–362.

[24] Hornung, R. and Keasler, J. (2014). The RAJA portability layer: Overview and status. Technical Report LLNL-TR-661403, Lawrence Livermore National Laboratory, Livermore, USA.

[25] Kershaw, D. (1981). Differencing of the diffusion equation in Lagrangian hydrodynamic codes. *Journal of Computational Physics*, 39:375–395.

[26] Khairallah, S. and Anderson, A. (2014). Mesoscopic simulation model of selective laser melting of stainless steel powder. *Journal of Materials Processing Technology*, 214:2627–2636.

[27] Knoll, D. A. and Keyes, D. (2004). Jacobian-free Newton-Krylov methods: A survey of approaches and applications. *Journal of Computational Physics*, 193:357–397.

[28] Landau, L. and Lifschitz, E. (1988). *Hydrodynamics, Theoretical Physics*, volume VI. Nauka, Moscow, 4th edition.

[29] Li, B. (2005). *Discontinuous Finite Elements in Fluid Dynamics and Heat Transfer*. Springer.

[30] Lo, M. and van Leer, B. (2009). Analysis and implementation of Recovery-based Discontinuous Galerkin for diffusion. In *19th AIAA Computational Fluid Dynamics Conference*, San Antonio, Texas, USA. AIAA 2009-3786.

[31] Luo, H., Baum, J., and Löhner, R. (2006). A fast, *p*-Multigrid Discontinuous Galerkin method for the Euler equations on unstructured grids. *Journal of Computational Physics*, 211(2):767–783.

[32] Luo, H., Baum, J., and Löhner, R. (2008). A Discontinuous Galerkin method based on a Taylor basis for the compressible flows on arbitrary grids. *Journal of Computational Physics*, 227(20):8875–8893.

[33] Luo, H., Luo, L., and Nourgaliev, R. (2012a). A Reconstructed Discontinuous Galerkin method for the Euler equations on arbitrary grids. *Communication in Computational Physics*, 12(5):1495–1519.

[34] Luo, H., Luo, L., Nourgaliev, R., and Cai, C. (2011). A parallel, Reconstructed Discontinuous Galerkin method for the compressible flows on arbitrary grids. *Communication in Computational Physics*, 9(2):363–389.

[35] Luo, H., Luo, L., Nourgaliev, R., and Mousseau, V. (2009). A Reconstructed Discontinuous Galerkin method for the compressible Euler equations on arbitrary grids. In *19th AIAA Computational Fluid Dynamics Conference*, San Antonio, Texas, USA. AIAA 2009-3788.

[36] Luo, H., Luo, L., Nourgaliev, R., and Mousseau, V. (2010a). A Reconstructed Discontinuous Galerkin method for the compressible Navier-Stokes equations on arbitrary grids. *Journal of Computational Physics*, 229:6961–6978.

[37] Luo, H., Luo, L., Nourgaliev, R., and Mousseau, V. A. (2010b). A parallel Reconstructed Discontinuous Galerkin method for compressible flows on arbitrary grids. AIAA-2010-0366.

[38] Luo, H., Luo, L., Nourgaliev, R., and Mousseau, V. A. (2010c). A Reconstructed Discontinuous Galerkin method for the compressible Navier-Stokes equations on arbitrary grids. AIAA-2010-0364.

[39] Luo, H., Xia, Y., Li, S., Nourgaliev, R., and Cai, C. (2012b). A Hermite WENO Reconstruction-based Discontinuous Galerkin method for the Euler equations on tetrahedral grids. *Journal of Computational Physics*, 231:5489–5502.

[40] Luo, H., Xia, Y., Spiegel, S., Nourgaliev, R., and Jiang, Z. (2013). A Reconstructed Discontinuous Galerkin method based on a Hierarchical WENO reconstruction for compressible flows on tetrahedral grids. *Journal of Computational Physics*, 236:477–492.

[41] Nourgaliev, R. (1998). Modeling and analysis of heat and mass transfer processes during in-vessel melt progression stage of Light Water Reactor (LWR) severe accidents. Technical Report ISBN 91-7170-235-0, Royal Institute of Technology, Department of Nuclear Power Safety, Stockholm, Sweden. Doctoral Thesis.

[42] Nourgaliev, R., Christon, M., and Bakosi, J. (2014). Fully-implicit projection solver for fluid flows. Technical Report INL/EXT-13-28278, Idaho National Laboratory, Idaho Falls, USA.

[43] Nourgaliev, R., Dinh, N., and Youngblood, R. (2010a). Development, selection, implementation and testing of architectural features and solution techniques for Next Generation of System Simulation Codes to support safety case of the LWR life extension. Technical Report INL/EXT-10-19984, Idaho National Laboratory, Idaho Falls, USA.

[44] Nourgaliev, R., Luo, H., Schofield, S., Dunn, T., Anderson, A., Weston, B., and Delplanque, J.-P. (2015). Fully-Implicit Orthogonal Reconstructed Discontinuous Petrov-Galerkin Method for Multiphysics Problems. Technical Report LLNL-TR-664250, Lawrence Livermore National Laboratory, Livermore, USA.

[45] Nourgaliev, R., Park, H.-K., and Mousseau, V. A. (2010b). *Recovery Discontinuous Galerkin Jacobian-free Newton-Krylov Method for Multiphysics Problems*. Computational Fluid Dynamics Review 2010. World Scientific and Nature Publishing Group.

[46] Nourgaliev, R., Theofanous, T., Park, H., Mousseau, V., and Knoll, D. (2008). Direct Numerical Simulation of interfacial flows: Sharp-Interface Method (I-SIM). In *46th AIAA Aerospace Sciences Meeting and Exhibit*, Reno, Nevada, USA. AIAA-2008-1453.

[47] Oliveira, P. J. and Issa, R. I. (2001). An improved PISO algorithm for the computation of buoyancy-driven flows. *Numerical Heat Transfer, Part B*, 40:473–493.

[48] Park, H., Nourgaliev, R., Martineau, R., and Knoll, D. (2009). On physics-based preconditioning of the Navier-Stokes equations. *Journal of Computational Physics*, 228:9131–9146.

[49] Patankar, S. (1980). *Numerical Heat Transfer and Fluid Flow*. Taylor & Francis.

[50] Poulson, J., Marker, B., van de Geijn, R., Hammond, J., and Romero, N. (2013). Elemental: A new framework for distributed memory dense matrix computations. *ACM Transactions on Mathematical Software*, 39(2).

[51] Qui, J. and Shu, C.-W. (2003). Hermite WENO schemes and their application as limiters for Runge-Kutta discontinuous Galerkin method: one-dimensional case. *Journal of Computational Physics*, 193:115–135.

[52] Rebourcet, B. (2007). Some remarks on Kershaw's legacy diffusion scheme. In *Workshop on Numerical methods for multi-material fluid flows*, Prague, Czech Republic.

[53] Saad, Y. (2003). *Iterative Methods for Sparse Linear Systems*. SIAM, Philadelphia, 2nd edition.

[54] Saad, Y. and Schultz, M. (1986). GMRES: a Generalized Minimal Residual algorithm for solving linear systems. *SIAM Journal of Science and Statistical Computing*, 7:856.

[55] Toro, E. F. (1999). *Riemann solvers and numerical methods for fluid dynamics. A practical Introduction*. Springer, Berlin/Heidelberg, 2nd edition.

[56] van Leer, B. and Nomura, S. (2005). Discontinuous Galerkin for diffusion. In *17th AIAA Computational Fluid Dynamics Conference*, Toronto, Ontario, Canada. AIAA 2005-5108.

[57] Voller, V. and Prakash, C. (1987). A fixed grid numerical modelling methodology for convection-diffusion mushy region phase-change problems. *International Journal of Heat Mass Transfer*, 30(8):1709–1719.

[58] Xia, Y., Lio, X., Luo, H., and Nourgaliev, R. (2015). A third-order implicit Discontinuous Galerkin method based on Hermite WENO reconstruction for time-accurate solution of the compressible Navier-Stokes equations. *International Journal for Numerical Methods in Fluids*, 79:416–435.

[59] Xia, Y., Luo, H., Frisbey, M., and Nourgaliev, R. (2014a). A set of parallel, implicit methods for Reconstructed Discontinuous Galerkin method for compressible flows on 3D hybrid grids. *Computers & Fluids*, 96:406–421.

[60] Xia, Y., Luo, H., and Nourgaliev, R. (2014b). An Implicit Hermite WENO Reconstruction-based Discontinuous Galerkin on tetrahedral grids. *Computers & Fluids*, 98:134–151.

[61] Youngblood, R., Nourgaliev, R., Kelly, D., Smith, C., and Dinh, T.-N. (2011). Framework for applying a Next-Generation Safety Analysis Code to plant life extension decision-making. In *ANS Proceedings of the 2011 ANS Summer Meeting*, Hollywood, Florida.